



Serial No.: 10/723,292  
W&B Docket No.: INF 2067-US  
OC Docket No.: INFN/0042

# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:  
Nikolaus Bröls

Serial No.: 10/723,292

Filed: November 26, 2003

Confirmation No.: 5561

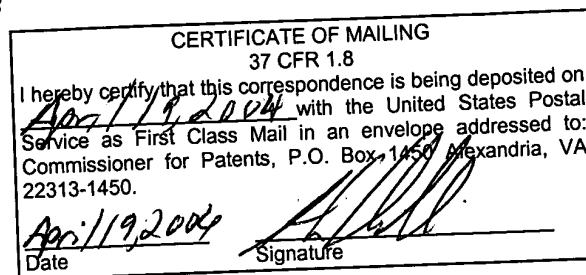
For: VITERBI DECODER

Group Art Unit: 2127

Examiner: UNKNOWN

MAIL STOP  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Dear Sir:



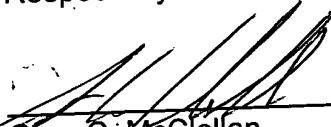
## CLAIM TO PRIORITY

Applicant(s) reaffirm the claim for the benefit of filing date of the following foreign patent application referred to in Applicant's Declaration:

German Patent Application Serial Number 102 55 426.9-35 filed November 28, 2002.

A copy of the application certified by the German Patent Office is enclosed.

Respectfully submitted,

  
Gero G. McClellan  
Registration No. 44,227  
MOSER, PATTERSON & SHERIDAN, L.L.P.  
3040 Post Oak Blvd. Suite 1500  
Houston, TX 77056  
Telephone: (713) 623-4844  
Facsimile: (713) 623-4846  
Agent for Applicant(s)



## Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

**Aktenzeichen:** 102 55 426.9

**Anmeldetag:** 28. November 2002

**Anmelder/Inhaber:** Infineon Technologies AG, München/DE

**Bezeichnung:** Viterbi-Decoder

**IPC:** H 03 M 13/41

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der  
ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 14. November 2003  
Deutsches Patent- und Markenamt  
Der Präsident  
Im Auftrag

A handwritten signature in black ink, appearing to be 'Wehner'.

Wehner

## Beschreibung

## Viterbi-Decoder

5

Die Erfindung betrifft einen Decoder zur Decodierung von Faltungscodes, gemäß dem Oberbegriff des Patentanspruchs 1. In der nachstehenden Beschreibung wird zum Stand der Technik Bezug genommen auf folgende Quellen aus der Fachliteratur:

10

- [1] G.D. Forney, jr. "The Viterbi Algorithm"; Proc. IEEE, Vol.61, No.3, March 1973, pp. 268-278.
- [2] P.Black et al. "A 140-Mb/s, 32-State, Radix-4 Viterbi Decoder"; IEEE Journal OF Solid-State Circuits, Vol.27, No.12, December 1992, pp. 1877-1885.
- [3] G. Fettweis et al. "A 100 Mbit/s Viterbi Decoder Chip: Novel Architecture and its Realization"; Proc. IEEE ICC, Vol.2, Atlanta August 1990, pp. 1877-1885.
- [4] A. Yeung et al. "A 210Mb/s Radix-4 Bit-level Pipelined Viterbi Decoder", 1995 IEEE International Solid-State Circuits Conference, pp. 88-89.
- [5] V.S. Gierenz et al. "A 550 Mb/s Radix-4 Bit-Level Pipelined 16-State 0.25- $\mu$ m CMOS Viterbi Decoder"; Proc. IEEE Inter.Conf. on Application-Specific Systems, Architectures, and Processors; Boston, July 2001, pp. 195-201.
- [6] R. Krambeck et al. "High-Speed Compact Circuits with CMOS"; IEEE Journal OF Solid-State Circuits, Vol.SC-17, No.3, June 1982, pp. 614-619.

15

20

25

30

Zur Übertragung einer Sequenz digitaler Daten von einer Nachrichtenquelle über einen gestörten Kanal an eine Nachrichtensenke werden in zunehmendem Maße sogenannte Faltungscodes verwendet. Ein Faltungscodierer verknüpft jedes Element oder "Wort" der Originalsequenz der Quelle gemäß einer gewählten Funktion mit einer bestimmten Anzahl der jeweils direkt vor-

35

angegangenen Wörter, um eine Sequenz von Sendewörtern auszugeben, die eine größere Bitbreite als die Wörter der Originalsequenz haben. Aufgrund dieser Redundanz in den Sendewörtern ist es möglich, in den empfangenen Wörtern auf der

5 Empfangsseite eventuelle Verfälschungen der Sendewörter zu erkennen und auch zu korrigieren.

Bei Kenntnis der gewählten Verknüpfungsfunktion, die den Code eindeutig definiert, lässt sich nämlich feststellen, inwie-

10 weit eine Gruppe mehrerer aufeinander folgender Empfangswörter in das Schema des Codes "passt". Hierzu muss ein Faltungsdecoder die Wortgruppe mit allen denjenigen "erlaubten" Wortgruppen vergleichen, die theoretisch (bei allen möglichen Originalsequenzen) hätten empfangen werden können, wenn der

15 Codierer einwandfrei arbeiten würde und die Übertragung störungsfrei wäre. Aus diesen Vergleichen werden sogenannte "Metriken" ermittelt, die angeben, wie nahe oder wie fern die Empfangswortgruppe zu den einzelnen erlaubten Wortgruppen liegt, also wie wahrscheinlich oder unwahrscheinlich es ist,

20 dass die Empfangswortgruppe aus der verglichenen erlaubten Wortgruppe hervorgegangen ist. In einer weiteren Vergleichoperation werden alle so ermittelten Metriken miteinander verglichen, und die erlaubte Wortgruppe, für welche die ermittelte Metrik die maximale Wahrscheinlichkeit angibt,

25 wird als gültige Wortgruppe ausgewählt.

Die Abstandsfunktion, die der Metrikberechnung zugrundeliegt, wird vorzugsweise den Eigenschaften des verwendeten Übertragungskanals und der verwendeten Modulationsform angepaßt.

30 Dies geschieht unter wahrscheinlichkeitstheoretischen Überlegungen. Die Abstandsfunktion kann so konzipiert sein, dass die Metrik umso kleiner ist, je näher sich die Vergleichsobjekte kommen. In diesem Fall wird die erlaubte Wortgruppe, für welche die Metrik minimal ist, als gültige Wortgruppe

35 ausgewählt; es erfolgt also eine Decodierung durch Minimum-Auswahl. Die Abstandsfunktion kann aber auch so konzipiert sein, dass die Metrik umso größer ist, je näher sich die

Vergleichsobjekte kommen. In diesem Fall wird die erlaubte Wortgruppe, für welche die Metrik maximal ist, als gültige Wortgruppe ausgewählt (Decodierung mit Maximum-Auswahl)

- 5 Von allen denkbaren Algorithmen zur Realisierung der vorstehend beschriebenen Faltungsdecodierung hat sich mittlerweile in der Praxis der sogenannte Viterbi-Algorithmus durchgesetzt, dessen allgemeines Prinzip z.B. in [1] beschrieben ist. Die vorliegende Erfindung bezieht sich auf die schaltungstechnische Ausbildung eines mit diesem Algorithmus arbeitenden Viterbi-Decoders. Zum besseren Verständnis der Erfindung wird nachstehend zunächst das Grundprinzip von Viterbi-Decodern und deren Aufbau nach dem Stand der Technik anhand der Figuren 1 bis 10 erläutert.
- 10
- 15
- Fig. 1 zeigt ein Diagramm der Abwicklung eines Radix-2-Trellis mit 4 Zuständen im Falle digitaler Übertragung;
- Fig. 2 zeigt ein Diagramm der Abwicklung eines Radix-2-Trellis mit 4 Zuständen im Falle analoger Übertragung;
- 20
- Fig. 3 zeigt die wichtigsten Teilbausteine eines Viterbi-Decoders;
- Fig. 4 zeigt die Transformation des Diagramms nach Fig. 1 in einen Radix-4-Trellis mit 4 Zuständen;
- 25
- Fig. 5 zeigt die Transformation des Diagramms nach Fig. 2 in einen Radix-4-Trellis mit 4 Zuständen;
- Fig. 6 zeigt ein Beispiel für einen Radix-4-Trellis mit 8 Zuständen;
- 30
- Fig. 7 zeigt das allgemeine Schema eines Viterbi-Decoders mit einem Radix-4-Trellis, der 8 Zustände hat;
- Fig. 8 zeigt in Blockdarstellung den allgemeinen Aufbau und die gegenseitige Verbindung von Prozessorelementen in einem mit Bit-Pipelining arbeitenden Radix-4-Viterbi-Decoder nach dem Stand der Technik;
- 35
- Fig. 9 zeigt etwas ausführlicher den Aufbau eines der Prozessorelemente nach Fig. 8;

Fig. 10 zeigt den kritischen Pfad zwischen Prozessorelementen, wie sie in Fig. 8 dargestellt sind.

Die Basis des Viterbi-Algorithmus bildet ein sogenannter  
5 "Trellis", der den verwendeten Faltungscode in Form eines Verzweigungsdiagramms eindeutig beschreibt. Die Fig. 1 zeigt einen Trellis für einen Faltungscode, der aus einer Originalsequenz, deren Elemente jeweils nur 1 Bit umfassen, eine Sendesequenz bildet, deren Elemente jeweils 2 Bits umfassen. Die  
10 vier möglichen Zustände (00, 01, 10, 11) der jeweils jüngsten beiden Bits der Originalsequenz werden als "Trelliszustände" bezeichnet und sind längs der Ordinate an den Koordinaten A, B, C, D aufgetragen. Längs der Abszisse sind Zeitmarken  $t_0$ ,  $t_1$  ... eines Taktes aufgetragen. Im Diagramm sind Knoten als  
15 Kreise gezeichnet, die eine Matrix aus Zeilen (in Richtung der Abszisse bzw. Zeitachse) und Spalten (in Richtung der Ordinate) bilden. Jeder Knoten steht für einen Trelliszustand, der durch die Zeilenposition (Ordinate) des Knotens im Diagramm bestimmt ist. Von jedem der vier Knoten einer Spalte  
20 führen zwei Zweige, die als Pfeile dargestellt sind, zu zwei Knoten der nächstfolgenden Spalte. Daher der Name "Radix-2"-Trellis.

Die Verknüpfungsvorschrift des Codes ist in Fig. 1 durch  
25 Dualzahlen (dünn geschrieben) an den Ursprüngen und Enden der Zweigpfeile angegeben. Die einstellige Dualzahl (0 oder 1) am Startpunkt eines Zweiges steht für den Binärwert eines neuen Bits der Originalsequenz und wird im folgenden als "Eingabezahl" bezeichnet. Die zweistellige Dualzahl am Ende des Zweiges  
30 gibt an, welcher neue Ausgabezustand (Binärwerte der beiden neuen Bits der Sendesequenz) sich infolge der neuen Eingabezahl der Originalsequenz ergibt; diese Zahl wird im folgenden als "Ausgabezahl" bezeichnet. Ist z.B. der Trelliszustand, also die jüngsten beiden Bits der Originalsequenz,  
35 gleich 00 (ein A-Knoten) und wird ein neues Bit mit dem Binärwert 1 hinzugegeben, dann ergibt sich der neue Trelliszustand 01 (Zweig mit der Eingabezahl 1 vom A-Knoten einer

Spalte zum B-Knoten der nächsten Spalte), und die neue Ausgabezahl wird 11, wie am Ende des betreffenden Zweiges eingetragen. Hat das neue Bit der Originalsequenz hingegen den Wert 0, so dass auch der neue Trelliszustand gleich 00 ist (Zweig vom A-Knoten einer Spalte zum A-Knoten der nächsten Spalte), dann ergibt sich die Ausgabezahl 00. In ähnlicher Weise lassen sich anhand der Beschriftungen in Fig. 1 auch die durch den Faltungscode bestimmten Beziehungen erkennen zwischen einerseits jedem der drei anderen möglichen Trelliszustände und dem neuen Bit der Originalsequenz und andererseits der resultierenden neuen Ausgabezahl.

Die Führung der Zweige zwischen den Knoten zweier benachbarter Spalten und die an den Zweigen eingetragenen Eingabezahlen und Ausgabezahlen sind an jedem Spaltenpaar gleich und beschreiben den Faltungscode eindeutig und vollständig. Daher genügt die Darstellung nur eines Spaltenpaares als Trellis zur Beschreibung des Codes. Die Fig. 1 zeigt jedoch mehr als zwei aufeinander folgende Spalten entlang der Zeitachse, also eine zeitliche Abwicklung des Trellis, um die Entwicklung der 2-Bit-Ausgabezahlen der Sendesequenz aus einer 1 Bit breiten Beispiels-Originalsequenz aufzuzeigen:

Als Beispiel sei eine Originalsequenz mit der Bitfolge 0-1-1-0 ... betrachtet, wie in der Tabelle über dem Diagramm dargestellt. Als definierte Anfangssituation sei angenommen, dass der Trelliszustand vor dem ersten Bit der Originalsequenz gleich 00 ist (A-Knoten in der ersten Spalte). Das erste Bit der betrachteten Originalsequenz ist 0, es führt zum neuen Trelliszustand 00 (A-Knoten in der zweiten Spalte) und zur ersten Ausgabezahl 00 für die Sendesequenz. Das zweite Bit der Originalsequenz ist 1, es führt zum neuen Trelliszustand 01 (B-Knoten in der dritten Spalte) und zur zweiten Ausgabezahl 11 für die Sendesequenz. Das dritte Bit der Originalsequenz ist ebenfalls 1, es führt zum neuen Trelliszustand 11 (D-Knoten in der vierten Spalte) und zur dritten Ausgabezahl 01 für die Sendesequenz. Das vierte Bit der Originalsequenz

ist 0, es führt zum neuen Trelliszustand 10 (C-Knoten in der fünften Spalte) und zur vierten Ausgabezahl 01 für die Sendesequenz. Aus der Originalsequenz 0-1-1-0-... wird also die codierte Sendesequenz 00-11-01-01-... gebildet, wie es in der

5 Tabelle über dem Diagramm eingetragen ist. Die bei den beschriebenen Codierungsschritten durchfahrenen Zweige bilden zusammengekommen einen "Pfad", der quer durch das Trellisdiagramm geht und charakteristisch für die Originalsequenz ist und der in Fig. 1 dick hervorgehoben ist.

10

Bei der Decodierung auf der Empfangsseite gilt es, diesen charakteristischen Pfad zu rekonstruieren, und zwar durch Analyse der empfangenen Version der Sendesequenz. Die Schritte des hierzu benutzten Viterbi-Algorithmus seien

15 nachstehend an dem in Fig. 1 gezeigten Codierungsbeispiel beschrieben.

20

Es sei vorerst angenommen, dass die Ausgabezahlen der Sendesequenz digital als 2-Bit-Wörter übertragen werden. Ein Übertragungsfehler kann sich also nur darin äußern, dass einzelne Bits in der Empfangssequenz invertiert sind. Beim Erscheinen jedes Empfangssymbols, also mit jedem 2-Bit-Wort der Empfangssequenz, wird die "Zweig"-Metrik aller (acht) Zweige zwischen benachbarten Spalten im Trellis nach Fig. 1 gegenüber diesem Empfangssymbol ermittelt. Basis hierfür bilden die in Fig. 2 am Ende der Zweige eingetragenen Ausgabezahlen. Wenn, wie im angenommenen Fall, Übertragungsfehler nur als Bit-Invertierung erscheinen können, kann es genügen, als Zweigmetrik die Anzahl der übereinstimmenden Bits in den

25 verglichenen Kandidaten zu ermitteln (entspricht 2 minus Hammingdistanz). Bei Übereinstimmung aller Bits ist die Zweigmetrik gleich 2, bei Übereinstimmung nur eines Bits ist die Zweigmetrik gleich 1, bei Nichtübereinstimmung aller Bits ist die Zweigmetrik gleich 0.

30

In der Fig. 1 sind die Zweigmetriken ZM als fett geschriebene Dezimalzahlen auf allen denjenigen Zweigen eingetragen, die

35



bei allen möglichen Empfangssequenzen durchlaufen werden könnten, wenn beim Trelliszustand 00 begonnen wird (A-Knoten in der ersten Spalte der Fig. 1), entsprechend der oben definierten Anfangssituation. Somit scheiden die Zweige aller  
5 derjenigen Pfade, die ausschließlich den Knoten B, C und D der ersten Spalte entspringen, von vorn herein aus. Das heißt, die betreffenden Zweige sind "unmöglich", und ihre Zweigmetriken müssen unberücksichtigt bleiben; sie sind daher in der Fig. 1 mit "X" bezeichnet.

10

Der erste Schritt auf der Empfangsseite, also die Verarbeitung des ersten Empfangssymbols, beginnt am A-Knoten der ersten Spalte. In der zweiten Spalte sind es nur die Knoten A und B, an denen es "mögliche" Eingangszweige gibt, und zwar  
15 jeweils nur einer. Die Metriken dieser Zweige werden als "Pfadmetriken" PM der betreffenden Knoten gespeichert, sie sind in der Fig. 1 als fette Dezimalzahlen innerhalb der die Knoten darstellenden Kreise eingetragen.

20

Bei jedem der folgenden Schritte werden für alle Knoten jeweils neu Pfadmetriken ermittelt und gespeichert. Zur Ermittlung der jeweiligen neuen Pfadmetriken, die innerhalb der die betreffenden Trellisknoten darstellenden Kreise eingetragen sind, werden zwei algebraische Summen ermittelt: für  
25 jeden der beiden Eingangszweige am Knoten die Summe der Zweigmetrik dieses Zweiges mit der Pfadmetrik des Ursprungsknotens dieses Zweiges. Die beiden Summen werden verglichen, und die größere von beiden wird als akkumulierte neue Pfadmetrik des betreffenden Knotens gespeichert. Nur der Eingangszweig, der diese neu Pfadmetrik hervorbrachte, bleibt  
30 als "Survivor" (Überlebender) des betreffenden Knotens für die Rekonstruktion des charakteristischen Pfades interessant.

35

Sofern an einem Knoten ein "unmöglicher" Eingangszweig mündet, muss dieser Zweig natürlich ohne Einfluss auf die Pfadmetrik-Berechnung bleiben. Dies kann man z.B. erreichen, indem man einfach die Pfadmetriken vor dem ersten Schritt

(also an den Knoten der ersten Spalte) so einstellt, dass die Pfadmetrik am Startknoten A der ersten Spalte um mindestens das Maß der maximal möglichen Zweigmetrik (also um mindestens die Zahl 2 im Falle der Fig. 1) größer ist als an den anderen Knoten der ersten Spalte. Wählt man z.B. für den Startknoten A der ersten Spalte die Pfadmetrik 0, wie in Fig. 1 dargestellt, dann sind die Pfadmetriken der anderen Knoten dieser Spalte auf die negative Zahl  $-(\geq 2)$  einzustellen. Alternativ kann man natürlich auch die Pfadmetrik des Startknotens auf  $\geq 2$  und die Pfadmetriken der anderen Knoten auf 0 einstellen.

Anhand der Eintragungen in den Trellisknoten der Fig. 1 erkennt man, dass sich der charakteristische Pfad vom Ende her rekonstruieren lässt, indem man jeweils von demjenigen Knoten einer Spalte, dessen Pfadmetrik am größten ist, zu demjenigen Knoten der vorangehenden Spalte zurückschreitet, der dort die maximale Pfadmetrik hat. Somit gibt die Sequenz der Eingabezahlen der Zweige dieses Pfades die Originalsequenz wieder.

Wegen der Redundanz der Codierung ist der charakteristische Pfad resistent gegenüber einem gewissen Maß an Übertragungsfehlern. Es sei z.B. angenommen, dass beim Empfang das erste Bit 0 des dritten Sendesymbol fälschlich als 1 angekommen ist (oder fälschlich als 1 bewertet wurde), wie in Klammern () in Fig. 1 gezeigt. Beim Decodierschritt 3 ergeben sich dann die in Klammern angegebenen Zweigmetriken, was zu den in Klammern angegebenen Änderungen der Pfadmetriken führt. Wie ersichtlich, führt die Rekonstruktion des charakteristischen Pfades jedoch zum gleichen Ergebnis wie bei fehlerfreier Übertragung. Die Originalsequenz wird also trotz des besagten Übertragungsfehlers korrekt wiedergewonnen.

In vielen Fällen wird die Sendesequenz nicht digital sondern in Form von Analogwerten übertragen. Das bedeutet für die in Fig. 1 gezeigte Originalsequenz 0-1-1-0, dass die am Faltungscodierer ausgegebenen Digitalwörter 00-11-01-01 in die entsprechenden Analogwerte 0-3-1-1 umgewandelt werden, um

z.B. eine Sendespannung innerhalb eines Bereichs zwischen 0 und 3 Volt zu modulieren, wie in Fig. 2 gezeigt. Auf der Empfangsseite wird diese Spannung in synchronisierter Weise abgetastet. Wegen unvermeidlicher Verzerrungen und anderer  
5 Störungen im Übertragungskanal werden die Abtastwerte nicht hochgenau den Werten der Sendesequenz entsprechen, sondern mehr oder weniger davon abweichen. In der Fig. 2 ist der Fall gezeigt, dass aus der Sendesequenz 0-3-1-1 die empfangsseitigen Abtastwerte 0,2--2,7--1,6--1,1 gewonnen werden.

10 Die Decodierung nach dem Viterbi-Algorithmus verläuft in diesem Fall sehr ähnlich wie im Falle der Fig. 1, nur dass die Pfadmetriken nicht aus dem Zweierkomplement der Hammingdistanz ermittelt werden, sondern arithmetisch. So können die  
15 Pfadmetriken z.B. bestimmt werden durch Ermittlung des Betrages der Differenz zwischen Ausgabezahl, die in Fig. 2 als Dezimalzahl geschrieben ist, und Empfangs-Abtastwert und Subtraktion dieses Betrages von irgendeinem festen Wert, der gleich oder größer ist als der höchstmögliche Abtastwert. In  
20 der Fig. 2 sind die so ermittelten Zweigmetriken ZM und auch die jeweils resultierenden maximalen Pfadmetriken PM für den Fall eingetragen, dass der besagte feste Wert gleich 3 ist. Es zeigt sich, dass der beim Decodieren vom Ende her zurückverfolgte Pfad über die jeweils größten Pfadmetriken (dick  
25 gezeichneter Linienzug) genau dem charakteristischen Pfad entspricht, mit dem die Originalsequenz exakt wiedergewonnen wird, trotz der von der Sendesequenz abweichenden Abtastwerte der Empfangssequenz.

30 Aus der vorstehenden Grundlagenbeschreibung ergibt sich, dass ein Viterbi-Decoder insgesamt die in der Fig. 3 als Blöcke gezeichneten Teilbausteine benötigt: erstens eine Zweigmetrik-Berechnungseinrichtung, gewöhnlich abgekürzt bezeichnet als BMU (Branch Metric Unit), welche die Eingangssequenz  
35 (Empfangssequenz) empfängt und mit dem Trellis des verwendeten Faltungscodes programmiert ist, um die Zweigmetriken im Trellis abhängig von der Eingangssequenz zu berechnen; zwei-

tens eine Addier-Vergleichs-Auswahl-Einrichtung, gewöhnlich abgekürzt bezeichnet als ACSU (Add-Compare-Select Unit), zur Berechnung der Pfadmetriken durch die oben beschriebenen Additions- und Vergleichsvorgänge und zum Treffen der Entscheidung bezüglich der Auswahl der jeweils maximalen Pfadmetrik für die Trellisknoten; drittens einen Pfadspeicher, gewöhnlich abgekürzt bezeichnet als SMU (Survivor Memory Unit), zur Speicherung der in der ACSU getroffenen Entscheidungen, welche die jeweiligen Survivoren bezeichnen, aus denen sich dann der charakteristische Pfad rekonstruieren lässt, um die Originalsequenz wiederzugewinnen.

In der BMU können die Zweigmetriken für alle Zustände A, B, C und D des Trellis, also an allen Knoten längs einer Spalte, parallel berechnet werden. In der ACSU wird jede neue Pfadmetrik ermittelt durch Auswahl der jeweils höchsten Summe von vorhergehender Pfadmetrik und neuen Zweigmetriken, wie es oben anhand der Fig. 1 beschrieben wurde. Somit braucht die ACSU eine Rückkopplungsschleife mit Speicherregister für die vorhergehende Pfadmetrik, wie in Fig. 3 gezeigt. Aufgrund dieser Rückkopplungsschleife ist die ACSU besonders kritisch für das Zeitverhalten des Decoders, was den Durchsatz begrenzt. Ein den Durchsatz steigerndes "Pipelining" (also eine Fließbandverarbeitung unter Einfügung von Pipeline-registern für Zwischenwerte) bei der Addition-Vergleich-Auswahl-Operation für die einzelnen Pfade ist in der ACSU nicht möglich wegen der Datenabhängigkeit aufeinander folgender Berechnungsschritte. Die Pfadmetriken für ein Eingangssymbol können nämlich erst berechnet werden, wenn die Pfadmetriken für das vorherige Eingangssymbol vorliegen. Eine weitere wichtige Randbedingung bei der Lösung des Durchsatzproblems stellt das eingeschränkte Energiebudget (Forderung nach möglichst wenig Energiebedarf z.B. wegen einfacher Plakette) und die Forderung nach möglichst kostengünstiger Implementierung in einem Standard-CMOS-Prozess dar. Deswegen kommt die zwar schnelle aber teure und stromfressende Bipolar- bzw. ECL-Technologie meist nicht in Betracht. Aus der

Literatur sind aber verschiedene architekturelle Ansätze bekannt, den Durchsatz des Decoders zu steigern.

Ein erster Ansatz zur Durchsatzsteigerung besteht darin,  
5 statt eines Radix-2-Trellis, wie er in Fig. 1 dargestellt ist  
(jeweils zwei Zweige am Eingang und Ausgang der Trellisknoten), einen Radix-4-Trellis zu benutzen, der jeweils vier  
Zweige am Eingang und Ausgang der Trellisknoten hat, wie es  
z.B. in [2] beschrieben ist. Die Fig. 4 zeigt die zeitliche  
10 Abwicklung eines Radix-4-Trellis, geltend für den gleichen  
Faltungscode und die gleichen Original- und Sendesequenzen  
wie in Fig. 1. Ein Vergleich der Figuren 1 und 4 zeigt, dass  
im Radix-4-Trellis jeweils zwei aufeinander folgende kleine  
Schritte des Radix-2-Trellis zu einem einzigen großen Schritt  
15 zusammengefasst sind. In Fig. 4 entsprechen die Knoten, welche  
für Beginn und Ende des großen Schrittes 1 gelten, denjenigen  
Knoten der Fig. 1, die dort für den Beginn des kleinen  
Schrittes 1 und das Ende des kleinen Schrittes 2 gelten. Die  
Knoten, die in Fig. 4 für Beginn und Ende des großen Schrit-  
20 tes 1 gelten, entsprechen denjenigen Knoten der Fig. 1, die  
dort für den Beginn des kleinen Schrittes 3 und das Ende des  
kleinen Schrittes 4 gelten.

Wegen der Zusammenfassung jeweils zweier Schritte ist gemäß  
25 Fig. 4 die Anzahl der Ausgangsverzweigungen und die Anzahl  
einemündenden Zweige an jedem Knoten von 2 auf 4 verdoppelt.  
Dementsprechend haben auch die Eingabezahlen, die an den  
Anfängen der Zweige eingetragen sind und den Zweig kennzeichnen,  
jeweils doppelt so viele Bits wie in Fig. 1, nämlich 2  
30 Bits. Die Ausgabезahlen, die in Fig. 4 an den Enden der  
Zweige eingetragen und für den Code charakteristisch sind,  
haben vier Bits. Bei jedem Decodierschritt werden vier auf-  
einander folgende Bits der Empfangssequenz mit den Ausgabe-  
zahlen aller Zweige verglichen, um die jeweilige Zweigmetrik  
35 zu ermitteln. Die Zweig- und Pfadmetriken und der charakteri-  
stische Pfad für die hier betrachtete Beispiels-Originalse-  
quenz sind auch in der Fig. 4 als fett geschriebene Dezimal-

zahlen bzw. dick hervorgehobener Linienzug dargestellt. Es zeigt sich auch hier, dass der beim Decodieren vom Ende her zurückverfolgte Pfad über die jeweils größten Pfadmetriken (dick gezeichneter Linienzug) genau dem charakteristischen Pfad entspricht, mit dem die Originalsequenz wiedergewonnen wird. Es lässt sich leicht nachvollziehen, dass dies auch beim Auftreten von Übertragungsfehlern gelten wird, genau so wie im Falle der Fig. 1.

10 In Fig. 5 ist der Radix-4-Trellis für den Fall dargestellt, dass die Übertragung durch Analogwerte erfolgt und die Empfangssequenz durch Abtastung des empfangenen Analogsignals erhalten wird, wie oben in Verbindung mit Fig. 2 beschrieben. Ähnlich wie bei dem Radix-4-Trellis nach Fig. 3 werden auch  
15 hier jeweils zwei aufeinander folgende Empfangssymbole in einem gemeinsam "großen" Schritt verarbeitet. Bei jedem Schritt ergeben sich die Zweigmetriken aus der Summe des Betrages der Differenz zwischen dem jeweils ersten Empfangssymbol  $e_0$  und dem höherwertigen Teil  $s_0$  der Ausgabezahl und  
20 des Betrages der Differenz zwischen dem jeweils zweiten Empfangssymbol  $e_1$  und dem niedrigerwertigen Teil  $s_1$  der Ausgabezahl (die Werte  $s_0$  und  $s_1$  sind in der Fig. 5 jeweils dezimal geschrieben). Besagte Summe wird von einem festen Betrag abgezogen, der beim dargestellten Beispiel gleich 6 gewählt ist, also doppelt so hoch wie in Fig. 2. Es ergeben sich somit die gleichen Pfadmetriken wie in Fig. 2.

Durch den Übergang vom Radix-2-Trellis auf einen Radix-4-Trellis beim Decodieren steht die doppelte Zeit für jeden  
30 Schritt zur Verfügung. Im Radix-4-Trellis gibt es für jeden Trelliszustand jedoch vier mögliche Vorgängerzustände, so dass der Vergleich der Pfadmetriken und die Auswahl der jeweils maximalen Pfadmetrik zum Herausfinden des wahrscheinlichsten Vorgängerzustandes und somit zur möglichst korrekten  
35 Rekonstruktion des charakteristischen Pfades komplizierter werden. Insgesamt resultiert jedoch ein Durchsatzgewinn.

Natürlich sind Faltungscodes und dazu passende Viterbi-Decoder nicht beschränkt auf nur 4 Trelliszustände A bis D. Die Fig. 6 zeigt ein Beispiel für einen Radix-4-Trellis mit acht Zuständen A bis H, wie er in [2] beschrieben ist. Ebenso sind Faltungscodes und Viterbi-Decoder mit noch mehr Trelliszuständen möglich, z.B. mit 16 oder 32 Trelliszuständen. In jedem Fall werden jedoch, solange es sich um einen "Radix-4"-Trellis handelt, für jeden Trelliszustand 4 Zweigmetriken verarbeitet. In der Fig. 6 sind die vier Eingabezahlen an den Ursprüngen der Zweige, die den Eingabezahlen in Fig. 4 oder Fig. 5 entsprechen, nicht eingetragen, um die Zeichnung nicht zu verwirren. Aus dem gleichen Grund sind in Fig. 6 auch die den Code beschreibenden Ausgabbezahlen an den Enden der Zweige weggelassen.

Der in Fig. 3 gezeigte allgemeine Aufbau eines Viterbi-Decoders ist in der Fig. 7 etwas ausführlicher dargestellt für den in Fig. 6 gezeigten Beispielsfall eines Radix-4-Trellis mit acht Trelliszuständen A, B, C, D, E, F, G, H. Die ACSU enthält dementsprechend acht Abteilungen A bis H, um für jeden Trelliszustand bei jedem Taktintervall  $t$  der Periodendauer  $T$  die maximale akkumulierte Pfadmetrik  $PMM(t)$  zu berechnen. Diese Berechnung erfolgt durch Addition jeder der vier Zweigmetriken  $ZM1$  bis  $ZM4$ , die für dieses Taktintervall von der BMU geliefert werden, mit der dem betreffenden Zweig zugeordneten maximalen Pfadmetrik  $PMM$  des vorhergehenden Berechnungsvorganges, gegenseitigen Vergleich der vier Additionsergebnisse und Auswahl des Maximums. In der Fig. 7 ist nur die Abteilung A (für den Trelliszustand A) detaillierter dargestellt. In dieser Abteilung wird die Zweigmetrik  $ZM1$  mit der maximalen Pfadmetrik  $PMM$  aus der gleichen Abteilung addiert;  $ZM2$  wird mit  $PMM$  aus der Abteilung C addiert;  $ZM3$  wird mit  $PMM$  aus der Abteilung G addiert, und  $ZM4$  wird mit  $PMM$  aus der Abteilung G addiert, wie es dem Trellisdiagramm nach Fig. 6 entspricht. In den anderen Abteilungen B bis H ist die Zuordnung zwischen den Zweigmetriken  $ZM1:4$  und den jeweils zu addierenden Pfadmetriken  $PMM$  natürlich anders als in der Ab-

teilung A, und zwar jeweils so, wie es das Trellisdiagramm vorschreibt.

Da die Werte der akkumulierten Pfadmetriken PM durch fortlaufendes Hinzaddieren der Zweigmetriken ZM von Takt zu Takt immer größer werden, ist der Wertebereich der Pfadmetriken PM größer als der Wertebereich der Zweigmetriken ZM. Demnach muß die Bitbreite "m" der Verarbeitungseinrichtungen in der ACSU an den Wertebereich der Pfadmetriken PM angepaßt sein. In der Fig. 7 ist die gleiche Zahl "m" für die Bitbreite sowohl der Pfadmetriken als auch der Zweigmetriken eingetragen. Das heißt, ein in Dualzahldarstellung binärcodierter Zweigmetriken-Wert ist durch Einfügung vorangestellter Nullbits auf die Bitbreite m gebracht.

In jeder der acht ACSU-Abteilungen A bis H wird der Wert des dort ermittelten Maximums im Register der betreffenden Abteilung gespeichert, als neue maximale Pfadmetrik PMM für den nächsten Schritt. Die Information darüber, welche der vier Pfadmetriken PM1 bis PM4 in einer Abteilung die jeweils maximale Pfadmetrik PMM ist, bezeichnet den Survivor für den betreffenden Trelliszustand. Diese acht Survivor-Informationen werden in der SMU gespeichert.

Ein besonderes Problem ergibt sich, wenn die in der ACSU verarbeiteten Größen vielstellige Zahlen sind. Auch bei analoger Übertragung und Abtastung der Empfangssequenz erfolgt in der Praxis die weitere Verarbeitung der Abtastwerte, also deren Vergleich mit den codebeschreibenden Ausgabezahlen und die anschließenden Metrik-Berechnungen, natürlich wieder mittels der Digitaltechnik. Deswegen müssen die Abtastwerte digitalisiert werden. Zur Erzielung hoher Genauigkeit ist eine hohe Auflösung der Digitalisierung erwünscht, was meist eine relativ große Bitbreite und somit eine große Stellenzahl bzw. Wortlänge der digitalisierten Zweigmetriken ZM und somit auch der akkumulierten Pfadmetriken bedeutet. Bei der Addition mehrstelliger Dualzahlen in der ACSU erfordert die "Carry-



Propagierung", also die Akkumulation und das Durchleiten der Carry-Bits (Übertrag-Bits) von der letzten Stelle LSB zur ersten Stelle MSB eigene Miniaturschritte. Dieses sogenannte "Ripple", verlängert den Additionsvorgang umso mehr, je  
5 größer die Wortlänge bzw. Bitbreite ist.

Ein bekannter Ansatz zur Reduzierung des erwähnten Ripple-Effektes und somit zur weiteren Steigerung des Durchsatzes von Viterbi-Decodern ist das in [3] beschriebene "Bit-Pipelining" mit "Carry-Save"-Darstellung, also eine Fließbandverarbeitung der Bits mit redundanten Darstellungen zur Einsparung des Übertrages. Hierbei erfolgen die Addition und der Vergleich in der ACSU bitweise beginnend mit dem MSB, also beginnend dem Bit höchsten Stellenwertes, auf der Grundlage  
10 redundanter Zahlendarstellung der Pfadmetrik, wobei die Addition ohne durchgehende Carry-Propagierung durchgeführt wird. Eine aus [4] bekannte Weiterentwicklung besteht darin, durch Umcodierung der redundanten Zahlendarstellung der Pfadmetriken die Maximum-Identifizierung auf eine einfache ODER-Verknüpfung zu reduzieren. Schließlich kann man, wie aus [5]  
15 bekannt, den Vergleich auf Bit-Ebene parallel durchführen, wobei alle vier Pfadmetriken PM1 bis PM4, die beim Radix-4-Trellis für einen Trelliszustand ermittelt werden, parallel mit den drei anderen Pfadmetriken verglichen werden. Dieser Vergleich kann lokal für jeden Zweig und somit sehr effizient und schnell erfolgen. Aufgrund der redundanten Zahlendarstellung wird die Entscheidung für jeden Zweig mit 2 Bits codiert: eines drückt eine "präliminäre" (also vorläufige) Entscheidung aus, und das andere drückt eine "finale" (also endgültige) Entscheidung aus.  
20  
25  
30

Die Kombination aller vier beschriebenen Maßnahmen zur Durchsatzsteigerung, nämlich

- i) Übergang auf Radix-4-Trellis,
- 35 ii) Pipelining mit redundanter Zahlendarstellung,
- iii) Maximum-Identifizierung durch ODER-Funktion,
- iv) paralleler Vergleich auf Bit-Ebene,

stellt zur Zeit den Stand der Technik bei der CMOS-Implementierung für höchste Durchsatz-Anforderungen bei gleichzeitig moderater Energieaufnahme dar. Die Fig. 8 veranschaulicht in einer Blockdarstellung, wie eine Abteilung der in Fig. 6 dargestellten ACSU gemäß diesem Stand der Technik ausgebildet ist, und zwar speziell am Beispiel der Abteilung A. Alle acht Abteilungen sind in gleicher Weise aufgebaut, sie unterscheiden sich nur darin, mit welchen der acht maximalen Pfadmetriken PMM die in ihnen verarbeiteten vier Zweigmetriken ZM jeweils addiert werden.

Gemäß der Fig. 8 ist für jede der vier Zweigmetriken ZM1 bis ZM4 eine Kaskade hintereinander geschalteter Prozessorelemente PE vorgesehen. Jedes Prozessorelement PE innerhalb der Kaskade berechnet eines der  $m$  Pfadmetrik-Bits aus dem zugeordneten Bit der betreffenden Zweigmetrik. Jede Kaskade enthält also  $m$  Prozessorelemente, so dass die Anordnung für jeden Trelliszustand insgesamt 4 mal  $m$  Prozessorelemente umfasst. Dargestellt sind nur zwei davon, deren eines einem beliebigen  $n$ -ten Bit mit dem Stellenwert  $2^n$  innerhalb des  $m$ -Bit-Wortes zugeordnet ist, und deren anderes dem  $(n+1)$ -ten Bit (nächsthöherer Stellenwert  $2^{n+1}$ ) zugeordnet ist. Jedes Prozessorelement enthält drei Funktionsblöcke: einen Additionsblock ADD, einen Schreibweise-Modifikationsblock MOD und einen Vergleicherblock VGL. Alle vier Prozessorelemente, die jeweils dem selben Bit-Stellenwert zugeordnet sind, haben über ein 8-adriges Verteilungs-Leitungsbündel Verbindungen sowohl miteinander als auch mit 8 Eingängen eines Maximum-Identifizierungsblocks MAX.

In den folgenden Figuren und auch in der nachstehenden Beschreibung sind Prozessorelemente und die an der Verarbeitung beteiligten Bits durch Abkürzungen in Großbuchstaben mit einem nachgestellten Suffix bezeichnet. Im Suffix gibt eine arabische Ziffer 1, 2, 3 oder 4 (oder allgemein: "i") die Zuordnung zur jeweiligen Zweigmetrik ZM1, ZM2, ZM3 oder ZM4 (oder allgemein ZMi) an, und der Klammerausdruck gibt die

Zuordnung zum Bit-Stellenwert des betreffenden Zweigmetrik-Wortes an. Ein Doppelpunkt ":" zwischen zwei nachgestellten Suffix-Symbolen bedeutet "bis".

- 5    Aufbau und Wirkungsweise der Funktionsblöcke ADD, MOD und VGL sind in allen Prozessorelementen gleich. Die Fig. 9 zeigt detailliert, als stellvertretendes Beispiel für alle Prozessorelemente, den Aufbau speziell des Prozessorelementes PE4(n), also des Prozessorelementes für das n-te Bit der
- 10    Zweigmetrik ZM4, zusammen mit dem Funktionsblock MAX(n).

Im Betrieb arbeiten die Prozessorelemente einer Kaskade PE(MSB:LSB) zeitgestaffelt um einen Halbtakt  $T/2$  versetzt von links nach rechts, also beginnend mit dem MSB (höchstwertiges Bit) und fortschreitend bis zum LSB (niedrigstwertiges Bit).

15    In jedem Prozessorelement PE(n) empfängt der Block ADD das zugeordnete Bit n der Zweigmetrik ZMi und eine redundante 2-Bit-Darstellung PMMa:b(n) für das Bit n der maximalen Pfadmetrik. Der Block ADD liefert ein Summenbit  $SU_i(n)$ , das im

20    Halbtakt  $\Phi A$  über eine Latch-Schaltung  $L\Phi A$  weitergegeben wird, und ein Carry-Bit  $CA_i(n)$ , das zum vorangehenden Prozessorelement PE(n+1) der selben Reihe rückgekoppelt wird.

Der Block MOD hat einen Summenbit-Eingang zum Empfang des im Halbtakt  $\Phi A$  weitergegebenen Summenbits SU, einen Carry-Eingang, auf den das Carry-Bit vom Block ADD des nachfolgenden Prozessorelementes der selben Kaskade rückgekoppelt wird, und außerdem zwei Entscheidungsbit-Eingänge zum Empfang eines präliminären Entscheidungsbits EBP(n+1) und eines finalen Entscheidungsbits EBF(n+1), die vom Block VGL des vorangehenden Prozessorelementes PE(n+1) erzeugt werden. Der Block MOD, der aus einem System von UND-Gattern und ODER-Gattern besteht, verknüpft diese vier empfangenen Bits, um vier Ausgangsbits zu erzeugen: eine redundante präliminäre 2-Bit-

25    Darstellung PMPa:b für das Bit n der Pfadmetrik und eine

30    redundante finale 2-Bit-Darstellung PMFa:b für das Bit n der Pfadmetrik. Hierbei gelten folgende Wahrheitstabellen:

35

Tabelle 1

CA(n-1)	x	0	0	1	1
SU(n)	x	0	1	0	1
EBP(n+1)	0	1	1	1	1
PMPa(n)	0	0	1	1	1
PMPb(n)	0	0	0	0	1

Tabelle 2

CA(n-1)	x	0	0	1	1
SU(n)	x	0	1	0	1
EBF(n+1)	0	1	1	1	1
PMFa(n)	0	0	1	1	1
PMFb(n)	0	0	0	0	1

Hierbei steht x für "beliebiger Wert". Die vier MOD-Ausgangs-  
bits PMPa:b und PMFa:b werden über die Latch-Schaltungen LΦB  
im Halbtakt ΦB weitergegeben. Die Latch-Schaltungen LΦA und  
LΦB haben somit die Funktion von Pipeline-Registern.

Im einzelnen werden die präliminären Pfadmetrik-Bits PMPa:b  
vom Block MOD im Halbtakt ΦB auf ein zugeordnetes Adernpaar  
des 8-adrigen Verteilungs-Leitungsbündels gegeben. Dieses  
Verteilungsbündel enthält insgesamt vier verschiedene Adern-  
paare, deren jedes genau einem der vier Prozessorelemente  
PE1:4(n) zugeordnet ist.

Die finalen Pfadmetrik-Bits PMFa:b werden im Halbtakt ΦB dem  
Vergleicherblock VGL angelegt. Im selben Halbtakt empfängt  
der VGL-Block das Ergebnis einer ODER-Verknüpfung der präli-  
minären Pfadmetrik-Bits PMPa der drei anderen Prozessorele-  
menten PE und das Ergebnis einer ODER-Verknüpfung der präli-  
minären Pfadmetrik-Bits PMPb der drei anderen Prozessorele-  
menten PE, wobei besagte Bits von den betreffenden Adernpaa-  
ren des Verteilungsbündels abgeleitet werden. Der Block VGL  
empfängt im Halbtakt ΦB ferner das präliminäre Entscheidungs-  
bit EBP(n+1) des vorangehenden Prozessorelementes PE(n+1).  
Der VGL-Block enthält ein System von Multiplexern, um das  
präliminäre Entscheidungsbit EBP(n) und das finale Entschei-  
dungsbit EBF(n) zu erzeugen.

Der Block MAX enthält zwei ODER-Gatter, um die beiden Bits

PMMA:b(n) für die redundante 2-Bit-Darstellung des Bits n der maximalen Pfadmetrik zu erzeugen. Das eine ODER-Gatter empfängt alle vier präliminären Pfadmetrik-Bits PMP1:4a(n) aus den vier Prozessorelementen PE1:4(n) und stellt das Bit

5 PMMA(n) genau dann auf den Wert Binärwert 1, wenn mindestens eines dieser vier empfangenen Bits gleich 1 ist. Das andere ODER-Gatter empfängt alle vier präliminären Pfadmetrik-Bits PMP1:4b(n) aus den vier Prozessorelementen PE1:4(n) und stellt das Bit PMMb(n) genau dann auf den Wert Binärwert 1,

10 wenn mindestens eines dieser vier empfangenen Bits gleich 1 ist.

Das vorstehend anhand der Figuren 8 und 9 beschriebene ACSU-System nach dem Stand der Technik arbeitet in jeder der Ab-

15 teilungen A bis H wie folgt, um für jeden Trelliszustand A bis H aus den vier m-Bit-Wörtern der Eingangs-Pfadmetriken ZM1:4 und aus der Information der bis dahin akkumulierten alten Pfadmetrik PPM die Entscheidung über die neue maximale Pfadmetrik abzuleiten:

20 Vor dem Beginn des Betriebs erfolgt eine Initialisierung, indem alle Entscheidungsbits EBPi und EBFi und alle Bits für die maximalen Pfadmetriken PMMia:b in allen Prozessorelementen auf 0 gesetzt werden. Dann werden die aufeinander folgenden m-Bit-Wörter der von der BMU gelieferten Zweigmetriken ZM

25 in Zeitabständen T angelegt, und zwar jedes Wort ZMi in einem "zeitgestaffelten Parallelformat", wobei die Bits innerhalb des selben Wortes um einen Halbtakt-Abstand  $T/2$  zueinander versetzt sind, gemäß dem nachstehend tabellierten Schema:

Tabelle 3

$\Phi A$	$\Phi B$	$\Phi A$	$\Phi B$	$\Phi A$	
Wort 1 $ZMi(m-1)$		Wort 2 $ZMi(m-1)$		Wort 3 $ZMi(m-1)$	...
	Wort 1 $ZMi(m-2)$		Wort 2 $ZMi(m-2)$		...
		Wort 1 $ZMi(m-3)$		Wort 2 $ZMi(m-3)$	...
			Wort 1 $ZMi(m-4)$		...
				Wort 1 $ZMi(m-5)$	...
					...

Somit werden die  $m$  Bits  $2^{m-1}$ ,  $2^{m-2}$ ,  $2^{m-3}$ , ...,  $2^0$  eines jeden Zweigmetrik-Wortes nacheinander im Halbtakt-Abstand  $T/2$  in die Kaskade getaktet, beginnend mit dem MSB, und durchlaufen die so gebildete Pipeline mit der Taktfrequenz  $2/T$ . Mit dem Erscheinen des Bits  $2^{m-1}$  (also des MSB) des ersten Zweigmetrik-Wortes, werden alle Entscheidungsbits der ersten Kaskadenstufe, also die Bits  $EBPi(m-1)$  und  $EBFi(m-1)$ , auf den Binärwert 1 gestellt und während der gesamten Folgezeit auf diesem Wert gehalten.

In jedem Prozessorelemente  $PEi(n)$  wird im ADD-Block das Zweigmetrik-Bit  $ZMi(n)$  zu der 2-Bit-Zahl  $PMMa(n)$ ,  $PMMb(n)$  der maximalen Pfadmetrik  $PMM(n)$  aus der jeweils zuständigen ACSU-Abteilung addiert, das Maximum aus den vier neuen Pfadmetriken wird ausgewählt, und für jeden der vier Zweige werden die beiden Entscheidungsbits  $EBPi(q)$  und  $EBFi(q)$  gebildet. Das Zweigmetrik-Bit  $ZMi(n)$  deckt dabei den Wertebereich  $(0:1) \cdot 2^n$  ab (das Symbol  $*$  ist das Multiplikationszeichen), und die 2-Bit-Zahl  $PMMa:b(n)$  ist eine modifizierte Darstellung des zugeordneten Bits der neuen maximalen Pfadmetrik; diese Zahl deckt den Wertebereich  $(0:2) \cdot 2^n$  ab. Der Ausgang des ADD-Blockes deckt den Wertebereich  $(0:3) \cdot 2^n$  ab, wobei das Carry-Bit  $CAi(n)$  mit dem Wert  $2 \cdot 2^n$  zum vorangehenden Prozessorelement  $PE(n+1)$  weitergereicht wird. Am Eingang des MOD-Blockes liegt somit wieder eine Pfadmetrikbit-Darstellung

mit dem Wertebereich  $(0:2) \cdot 2^n$  an, gebildet durch das Summenbit  $SU(n)$  und das Carry-Bit  $CA(n-1)$  des nachfolgenden Prozessorelementes  $PE(n-1)$ .

- 5 Der MOD-Block modifiziert diese Zahlendarstellung wie folgt:

Tabelle 4

Eingangsinformation von MOD				Modifizierte Darstellung	
Zahlenwert		Eingangsbits			
dezimal	dual	SU	CA	a	b
0	00	0	0	0	0
1	01	0	1	1	0
		1	0		
2	10	1	1	1	1

Mit der so modifizierten Darstellung ist es möglich, durch die beschriebenen zwei 4-ODER-Verknüpfungen im MAX-Block die redundante 2-Bit-Darstellung  $PMMA:b(n)$  des Maximums der insgesamt vier präliminären 2-Bit-Pfadmetrikdarstellungen  $PMP1a:b(n)$ ,  $PMP2a:b(n)$ ,  $PMP3a:b(n)$  und  $PMP4a:b(n)$  zu gewinnen.

Die Entscheidungsbits  $EBPi(n)$  und  $EBFi(n)$  werden in den Blöcken MOD und VGL gemäß folgender Logikvorschrift gebildet:

Tabelle 5

$EBFi(n+1)$	$EBPi(n+1)$	$DIFFi(n)$	$EBFi(n)$	$EBPi(n)$
1	1	-2	0	0
		-1	1	0
		$\geq 0$	1	1
1	0	$\leq 0$	0	0
		+1	1	0
		+2	1	1
0	0	X	0	0

Hierin bedeutet  $DIFFi(n)$  die maximale Differenz zwischen der finalen Pfadmetrik  $PMFi(n)$ , die durch die 2 Bits  $PMFia:b(n)$  dargestellt wird, und den 3 präliminären Pfadmetriken  $PMj(n)$ ,

die aus den drei anderen Prozessorelementen  $PE_j(n)$  kommen und deren jede dargestellt werden durch die 2 Bits  $PMP_{ja:b}(n)$ , wobei  $j = 1, 2, 3, 4$  außer  $i$  ist.  $X$  bedeutet "beliebiger Wert".

5

Hat  $EBFi(n)$  den Wert 0, dann wird dies gewertet als Information zum definitiven Ausschluss des betreffenden Zweiges  $i$ . Die Kombination  $EBFi(n)=1$  und  $EBPi(n)=0$  wird gewertet als Information zu einem voraussichtlichen Ausschluss. Die Kombination  $EBFi(n)=1$  und  $EBPi(n)=1$  besagt, dass der betreffende  
10 Zweig vermutlich der Survivor sein wird. Die präliminären und finalen Entscheidungsbits werden beim MSB der Zeigmetrik (also beim  $(m-1)$ -ten Bit) beginnend berechnet und weitergereicht, bis schließlich am LSB (also beim 0-ten Bit) die eindeutige Entscheidung bereitsteht.  
15

Wie den Figuren 8 und 9 zu entnehmen ist, liegen zwischen den Gattern relativ lange Leitungen für die Verteilung der Signale, so dass entsprechende Leitungstreiber benötigt werden.

20 Diese Treiber sind in den Figuren 8 und 9 aus Gründen der Übersichtlichkeit nicht dargestellt, wohl aber in der Fig. 10, die den "kritischen Pfad" zeigt, das heißt, den zeitlich längsten Signalweg von einem Register (bzw. Latch) ins nächste. Dieser Signalweg ist es somit, der die höchstmögliche Geschwindigkeit des Gesamtsystems begrenzt. Geschwindigkeitssteigerungen (Taktfrequenzerhöhungen) können nur durch Beschleunigung dieses kritischen Pfades erzielt werden, bis schließlich ein anderer Pfad der kritischste ist.  
25

30 Obwohl die in den Figuren 8 und 9 gezeigte bekannte ACSU-Architektur bezüglich Durchsatz und Verlustleistung das derzeitige Optimum darstellt, können damit die erhöhten Anforderungen an fortgeschrittene Datenübertragungssysteme nicht befriedigend erfüllt werden. Dies gilt insbesondere für die  
35 in Entwicklung befindliche neue Generation von Festplatten-Controllern, die mit Datenraten von mehr als 2000 MBit/s arbeiten sollen.



Bei der vorstehenden Beschreibung des Standes der Technik wurde als Beispiel davon ausgegangen, dass die Metriken auf einer Abstandsfunktion basieren, bei welcher die Decodierung durch Maximum-Auswahl erfolgen muß. Die angesprochenen Probleme sind jedoch die gleichen, wenn man eine Abstandsfunktion benutzt, bei welcher statt der Maximum-Auswahl eine Minimum-Auswahl zu erfolgen hat. In jedem Fall gilt es, für jeden Trelliszustand aus mehreren akkumulierten Metriken die in einem vorbestimmten Sinn "extreme" Metrik herauszufinden, wobei dieser vorbestimmte Sinn, je nach Art der gewählten Abstandsfunktion, entweder auf das Maximum oder das Minimum zielt.

Die Aufgabe der Erfindung besteht darin, einen mit Bit-Pipelining arbeitenden Viterbi-Decoder so auszubilden, dass er zu höheren Durchsätzen fähig ist, als sie der bisherige Stand der Technik erlaubt. Diese Aufgabe wird erfindungsgemäß durch die im Patentanspruch 1 angegebenen Merkmale gelöst.

Demnach wird die Erfindung realisiert in einem Viterbi-Decoder, der ausgelegt ist zum Decodieren eines Faltungscodes auf der Basis eines Radix- $2^x$ -Trellis mit  $2^z$  Zuständen, wobei  $x$  und  $z$  ganze Zahlen  $\geq 1$  sind und wobei der Decoder im wesentlichen drei Berechnungseinheiten enthält. Die erste Berechnungseinheit, ermittelt aus periodisch aufeinander folgenden Mehrbit-Wörtern einer Eingangssequenz für jeden Trelliszustand die  $2^x$  Zweigmetriken als Mehrbit-Wörter. Die zweite Berechnungseinheit, die sogenannte ACSU, addiert für jeden Trelliszustand in jeder Taktperiode die jeweils zugeordneten  $2^x$  Zweigmetriken mit bisher akkumulierten  $2^x$  Pfadmetriken aus  $2^x$  Vorgänger-Trelliszuständen und wählt die in einem vorgewählten Sinne extreme der so ermittelten  $2^x$  Pfadmetriken als neue akkumulierte Pfadmetrik für den Additionsvorgang der nächsten Periode aus. Die dritte Berechnungseinheit ist ausgelegt zum Speichern der aus der zweiten Berechnungseinheit gewonnenen Informationen über die Identität der

Pfadmetriken, die in den aufeinander folgenden Taktperioden in der zweiten Berechnungseinheit ausgewählt werden. Die zweite Berechnungseinheit enthält für jeden Trelliszustand  $2^x$  parallele Kaskaden von Prozessorelementen, welche hintereinander geschaltet sind für eine Pipeline-Verarbeitung der Bits der Zweigmetriken und der extremen Pfadmetriken, fortschreitend mit abnehmendem Stellenwert der Bits. Ferner enthält die zweite Berechnungseinheit für jeden Trelliszustand jeweils eine Extremwert-Auswahleinrichtung für alle Prozessorelemente gleicher Ordnungszahl innerhalb der Pipeline. Die Erfindung ist dadurch gekennzeichnet, dass die Anzahl der Prozessorelemente in jeder der Kaskaden kleiner ist als die Anzahl  $m$  der Bits, die für die Dualzahldarstellung des Wertebereiches der Pfadmetriken verwendet werden, indem jede der Kaskaden jeweils eine dem ganzzahligen Anteil von  $m/p$  entsprechende Anzahl  $\text{INT}(m/p)$  von Prozessorelementen enthält, wobei  $p$  eine ganze Zahl mindestens gleich 2 und kleiner als  $m$  ist und wobei jedes dieser  $\text{INT}(m/p)$  Prozessorelemente innerhalb der Kaskade genau einem von  $\text{INT}(m/p)$  aufeinander folgenden disjunkten  $p$ -Bit-Gruppen der  $m$  Bits zugeordnet ist.

Der Quotient  $m/p$  kann ganzzahlig oder nicht-ganzzahlig sein. Im ersteren Fall ist  $\text{INT}(m/p)$  gleich  $m/p$ , und jede Kaskade braucht nur aus insgesamt  $m/p$  Prozessorelementen zu bestehen, deren jedem eine  $p$ -Bit-Gruppe zugeordnet ist, also jeweils eines der  $m/p$  aufeinander folgenden disjunkten " $p$ -Tupel" der  $m$  Bits. Eine Ganzzahligkeit von  $m/p$  kann man stets dadurch erreichen, dass man bei gegebener Zahl  $p$  die verwendete Bitbreite  $m$  für die Pfadmetriken im Bedarfsfall durch Voranstellung von Nullbits auf ein ganzzahliges Vielfaches von  $p$  bringt. Man kann es aber auch bei einer eventuellen Nicht-Ganzzahligkeit des Quotienten  $m/p$  belassen. In diesem Fall ist in jeder Kaskade zusätzlich zu den  $\text{INT}(m/p)$  Prozessorelementen, die jeweils einer  $p$ -Bit-Gruppe zugeordnet sind, ein weiteres Prozessorelement vorzusehen, das den restlichen Bits zugeordnet ist, deren Anzahl  $r$  kleiner ist als  $p$ . Dieses Prozessorelement wird vorzugsweise den  $r$  höchsten oder den  $r$

niedrigsten Stellenwerten in der m-Bit-Folge zugeordnet.

Die "im vorgewählten Sinne extremen" Metriken können entweder die jeweils maximalen oder die jeweils minimalen Metriken sein, je nachdem, welche Art von Abstandsfunktion für die Ermittlung der Zweigmetriken verwendet wird.

Mit der Erfindung wird also eine gegenüber dem Stand der Technik modifizierte Architektur der ACSU angegeben. Statt der bekannten 1-Bit-Prozessorelemente werden Mehrbit-Bit-Prozessorelemente verwendet, also p-Bit-Prozessorelemente mit  $2 \leq p < m$ , vorzugsweise mit  $p=2$ . Dies scheint zunächst dem Erfolg zu widersprechen, den man sich allgemein von dem Prinzip des Bit-Pipelining verspricht. Es zeigt sich jedoch, dass man mit der erfindungsgemäßen Mehrbit-Architektur tatsächlich eine Durchsatzsteigerung und zusätzlich noch weitere Vorteile erzielen kann. Bei einer erfindungsgemäßen Mehrbit-Architektur sind für eine gegebene Bitbreite m nur  $1/p$  so viele Prozessorelemente erforderlich wie bei der bekannten 1-Bit-Architektur. Hierdurch vermindert sich die Latenzzeit der Pipeline auf  $1/p$ -tel im Vergleich zur bekannten 1-Bit-Lösung. Ferner ist das zur Verfügung stehende Energiebudget pro Prozessorelement p-mal so hoch.

Allerdings wird durch die  $\geq 2$ -Bit-Architektur die Komplexität der Berechnungen erheblich gesteigert, was sich normalerweise negativ auf die Geschwindigkeit und den maximalen Durchsatz auswirken könnte. In besonderer Ausführungsform der Erfindung wird diesem eventuellen Nachteil jedoch entgegengewirkt durch Einführung einer neuen, redundanten Zahlendarstellung, womit es gelingt, den kritischen Pfad ähnlich kurz wie bei der bekannten Architektur zu halten, insbesondere wenn die Extremwert-Auswahl die Auswahl des Maximums ist. Hierbei ist allerdings wiederum die Komplexität der einzelnen Gatterstufen höher. Gemäß einer vorteilhaften Ausgestaltung der Erfindung kann jedoch eine signifikante Beschleunigung erzielt werden durch den Einsatz von dynamischen Schaltungstechniken wie der

aus der Literatur [6] bekannten DOMINO-Logik, ohne dass insgesamt ein Mehraufwand an Fläche und Energie erforderlich ist.

- 5 Die Erfindung wird nachstehend an einem Ausführungsbeispiel anhand der Figuren 11 bis 14 näher erläutert.

Fig. 11 zeigt in einer groben Blockdarstellung den Aufbau zweier aufeinander folgender Prozessorelemente einer erfindungsgemäß ausgebildeten ACSU mit  $p = 2$ , also mit einer 2-Bit-Architektur;

Fig. 12 zeigt eines der Prozessorelemente nach Fig. 11 in detaillierter Darstellung;

Fig. 13 zeigt isoliert die Bestandteile des kritischen Pfades in der erfindungsgemäß ausgebildeten ACSU;

Fig. 14 zeigt ein Beispiel für dynamische Schaltungstechnik.

Das im folgenden beschriebene Ausführungsbeispiel der Erfindung betrifft ebenso wie das oben beschriebene Beispiel des Standes der Technik den Aufbau einer ACSU-Abteilung für einen Radix-4-Trellis mit 8 Zuständen, um für einen Trelliszustand die neue maximale Pfadmetrik  $PMM(t+T)$  zu ermitteln aus den Zweigmetriken  $ZM1:4$  der vier Eingangszweige und den alten maximalen Pfadmetriken  $PMM(t)$  der den Eingangszweigen zugeordneten Vorgänger-Trelliszustände. Die in den Figuren 11 und 12 gezeigten Schaltungsblöcke ähneln in gewissen Aspekten den Blöcken des Standes der Technik nach den Figuren 8 und 9 und sind deswegen mit den gleichen Abkürzungen bezeichnet wie dort; gleiches gilt für die Signalbits und deren verwendete Kurzbezeichnungen.

Ein wesentlicher Unterschied gegenüber dem Stand der Technik besteht darin, dass bei dem beschriebenen Ausführungsbeispiel jedes Prozessorelement jeweils zwei Bits eines  $m$ -stelligen Pfadmetrik-Wortes verarbeitet; es ist also  $p = 2$ . Ferner wird davon ausgegangen, dass  $m/p$  ganzzahlig ist, d.h.  $m$  ist geradzahlig. Für jede der Pfadmetriken  $PM1:4$  sind demnach  $m/2$  Pro-

zessorelemente vorgesehen.  $1 \leq n \leq m$  sei die Ordnungszahl der aufeinander folgenden Bits, gezählt ab dem niedrigstwertigen Bit LSB, und  $1 \leq q \leq m/2$  sei die Ordnungszahl der direkt aufeinander folgenden disjunkten Bitpaare, gezählt ab dem niedrigstwertigen Bitpaar. Ein beliebiges Prozessorelement PE(q) verarbeitet das q-te Bitpaar, also das  $2n$ -te und das  $(2n+1)$ -te Bit. Das vorangehende Prozessorelement PE(q+1), sofern vorhanden, verarbeitet das (q+1)-te Bitpaar, also das  $(2n+2)$ -te und das  $(2n+3)$ -te Bit. Das nachfolgende Prozessorelement PE(q-1), sofern vorhanden, verarbeitet das (q-1)-te Bitpaar, also das  $(2n-2)$ -te und das  $(2n-1)$ -te Bit. In der nachstehenden Beschreibung und auch in den Figuren 11 und 12 ist das niedrigerwertige Bit jedes Bitpaars durch das Symbol "-0" und das höherwertige Bit durch das Symbol "-1" innerhalb des Suffix des Bitnamens gekennzeichnet.

Die insgesamt  $m/2$  mal 4 Prozessorelemente für die vier Zweigmetriken ZM1:4 haben alle den gleichen Aufbau und bilden eine Anordnung aus 4 Reihen. Die Fig. 11 zeigt, als stellvertretendes Beispiel für alle Prozessorelemente, zwei benachbarte Prozessorelemente PE4(q) und PE4(q+1) für das q-te und das (q+1)-te Bitpaar der Zweigmetrik ZM4, und zwar speziell für die ACSU-Abteilung A, die dem Trelliszustand A zugeordnet ist. Ferner gezeigt sind die Blöcke MAX für die Erzeugung der Bits, welche das Maximum PMM(q) bzw. PMM(q+1) der Pfadmetrik-Bits aus allen Prozessorelementen PE1:4(q) bzw. PE1:4(q+1) der ACSU-Abteilung A signalisieren. Die Fig. 12 zeigt detailliert den Aufbau der Prozessorelemente am Beispiel des Prozessorelementes PE4(q).

Im Betrieb arbeitet jede Reihe von  $m/2$  hintereinander geschalteten Prozessorelementen zeitgestaffelt um jeweils einen Halbtakt  $T/2$  versetzt von links nach rechts, also beginnend mit dem höchstwertigen Bitpaar und fortschreitend bis zum niedrigstwertigen Bitpaar.

In jedem Prozessorelement PE<sub>i</sub>(q) empfängt der Block ADD das

q-te Bitpaar  $Z_{Mi-0}(q)$ ,  $Z_{Mi-1}(q)$  der Zweigmetrik  $Z_{Mi}$  und eine redundante 4-Bit-Darstellung  $PMM_{ia:d}(q)$  des q-ten Bitpaars der maximalen Pfadmetrik. Der Block ADD liefert zwei Summenbits  $S_{Ui-0}$  und  $S_{Ui-1}$ , die im Halbtakt  $\Phi A$  weitergegeben werden, und ein Carry-Bit  $CA_i$ , das zum vorangehenden Prozessorelement  $PE(q+1)$  der selben Reihe rückgekoppelt wird. Der Block ADD enthält das in Fig. 12 gezeigte System von Gattern mit logischen Funktionen UND, NAND, ODER, Exklusiv-ODER (XOR) und Exklusiv-NOR (XNOR), um die Summe aus den Pfadmetriken  $PMM_{ia:d}(q)$ , die in 4-Bit-Thermometerdarstellung vorliegen, und den Zweigmetriken  $Z_{Mi-0:1}(q)$ , die als 2-Bit-Dualzahldarstellung vorliegen, zu bilden. Die Ausgänge  $S_{Ui-0}$ ,  $S_{Ui-1}$  und  $CA_i$  repräsentieren diese Summe in Dualzahldarstellung, wobei  $S_{Ui-0}$  die Wertigkeit  $2^{2n}$ ,  $S_{Ui-1}$  die Wertigkeit  $2^{(2n+1)}$  und  $CA_i$  die Wertigkeit  $2^{(2n+2)}$  besitzen.

Der Block MOD hat zwei Summenbit-Eingänge zum Empfang der im Halbtakt  $\Phi A$  weitergegebenen Summenbits  $SU-0$  und  $SU-1$ , einen Carry-Eingang, auf den das Carry-Bit vom Block ADD des nachfolgenden Prozessorelementes  $PE_i(q-1)$  rückgekoppelt wird, und außerdem zwei Entscheidungsbit-Eingänge zum Empfang eines präliminären Entscheidungsbits  $EBP(q+1)$  und eines finalen Entscheidungsbits  $EBF(q+1)$ , die vom Block VGL des vorangehenden Prozessorelementes  $PE(q+1)$  erzeugt werden. Der Block MOD, der aus einem System von UND-Gattern und ODER-Gattern besteht, verknüpft diese vier empfangenen Bits, um acht Ausgangsbits zu erzeugen: eine redundante präliminäre 4-Bit-Darstellung  $PMP_{a:d}$  für das q-te Bitpaar der Pfadmetrik und eine redundante finale 4-Bit-Darstellung  $PMF_{a:d}$  für das q-te Bitpaar der Pfadmetrik.

Die Logikfunktion des MOD-Blockes ist so, dass der Binärwert 0 eines Entscheidungsbits aus dem vorangehenden Prozessorelement  $PE_i(q+1)$  die jeweils zugeordnete Vierergruppe der MOD-Ausgangsbits des Prozessorelementes  $PE_i(q)$  "maskiert", also ebenfalls auf 0 setzt. Das heißt, wenn das präliminäre Entscheidungsbit  $EBP_i(q+1)$  den Binärwert 0 hat, dann gehen

alle vier Ausgangsbits  $PMPia:d(q)$  auf 0, und wenn das finale Entscheidungsbit  $EBFi(q+1)$  den Binärwert 0 hat, dann gehen alle vier Ausgangsbits  $PMFia:d(q)$  auf 0.

- 5 Die acht MOD-Ausgangsbits  $PMPia:d$  und  $PMFia:d$  werden über die dargestellten Latch-Schaltungen  $L\Phi B$  im Halbtakt  $\Phi B$  weitergegeben. Im einzelnen werden die vier präliminären Pfadmetrik-Bits  $PMPia:d$  vom Block MOD im Halbtakt  $\Phi B$  auf ein zugeordnetes Adernquartett eines 16-adrigen Verteilungs-Leitungsbündels gegeben. Dieses Verteilungsbündel enthält insgesamt vier  
10 verschiedene Adernquartette, deren jedes genau einem der vier Prozessorelemente  $PE1:4(q)$  zugeordnet ist.

- Die vier finalen Pfadmetrik-Bits  $PMFia:b(q)$  werden im  
15 Halbtakt  $\Phi B$  dem Vergleicherblock VGL angelegt. Im selben Halbtakt empfängt der VGL-Block das Ergebnis einer ODER-Verknüpfung der präliminären Pfadmetrik-Bits  $PMPja$  der drei anderen Prozessorelemente  $PEj(q)$ , das Ergebnis einer ODER-Verknüpfung der präliminären Pfadmetrik-Bits  $PMPjb$  der drei  
20 anderen Prozessorelemente  $PEj(q)$ , das Ergebnis einer ODER-Verknüpfung der präliminären Pfadmetrik-Bits  $PMPjc$  der drei anderen Prozessorelemente  $PEj(q)$  und das Ergebnis einer ODER-Verknüpfung der präliminären Pfadmetrik-Bits  $PMPjd$  der drei anderen Prozessorelemente  $PEj(q)$ . Der Block VGL empfängt im  
25 Halbtakt  $\Phi B$  ferner das präliminäre Entscheidungsbit  $EBPj(q+1)$  des vorangehenden Prozessorelementes  $PEj(q+1)$ . Der VGL-Block enthält ein System von Multiplexern, um das präliminäre Entscheidungsbit  $EBPj(q)$  und das finale Entscheidungsbit  $EBFj(q)$  zu erzeugen.

- 30 Der Block MAX enthält vier ODER-Gatter, um die vier Bits  $PMMia:d(q)$  für die redundante 4-Bit-Darstellung des  $q$ -ten Bitpaares der maximalen Pfadmetrik zu erzeugen. Das erste ODER-Gatter empfängt alle vier präliminären Pfadmetrik-Bits  
35  $PMP1:4a(q)$  aus den vier Prozessorelementen  $PE1:4(q)$  und stellt das Bit  $PMMa(q)$  genau dann auf den Wert Binärwert 1, wenn mindestens eines dieser vier empfangenen Bits gleich 1

ist. Das zweite, dritte und vierte ODER-Gatter verknüpfen nach der gleichen Regel die präliminären Pfadmetrik-Bits  $PMP1:4b(q)$  bzw.  $PMP1:4c(q)$  bzw.  $PMP1:4d(q)$  aus den vier Prozessorelementen  $PE1:4(n)$ .

5

Das vorstehend anhand der Figuren 11 und 12 beschriebene neue ACSU-System arbeitet wie folgt, um für jeden Trelliszustand A bis H aus den vier m-Bit-Wörtern der Eingangs-Pfadmetriken  $ZM1:4$  und aus der Information der bis dahin akkumulierten alten Pfadmetrik  $PMM$  die Entscheidung über die neue maximale Pfadmetrik abzuleiten:

10

Vor dem Beginn des Betriebs erfolgt eine Initialisierung, indem alle Entscheidungsbits  $EBP_i$  und  $EBF_i$  und alle Bits für die maximalen Pfadmetriken  $PMM_{ia:d}$  in allen Prozessorelementen auf 0 gesetzt werden. Dann werden aufeinander folgende m-Bit-Wörter der von der BMU gelieferten Zweigmetriken  $ZM$  in Zeitabständen  $T$  angelegt, und zwar derart, dass aufeinanderfolgende disjunkte Bitpaare innerhalb des selben Wortes um einen Halbtakt-Abstand  $T/2$  zueinander versetzt sind, gemäß dem nachstehend tabellierten Schema:

15

20



Tabelle 6

$\Phi A$	$\Phi B$	$\Phi A$	$\Phi B$	$\Phi A$	
Wort 1 $ZMi(m-1)$ $(m-2)$		Wort 2 $ZMi(m-1)$ $(m-2)$		Wort 3 $ZMi(m-1)$ $(m-2)$	...
	Wort 1 $ZMi(m-3)$ $(m-4)$		Wort 2 $ZMi(m-3)$ $(m-4)$		...
		Wort 1 $ZMi(m-5)$ $(m-6)$		Wort 2 $ZMi(m-5)$ $(m-6)$	...
			Wort 1 $ZMi(m-7)$ $(m-8)$		...
				Wort 1 $ZMi(m-9)$ $(m-10)$	...
					...

Somit werden die  $m/2$  Bitpaare  $(2^{m-1}, 2^{m-2})$ ;  $(2^{m-3}, 2^{m-4})$ ;  $(2^{m-5}, 2^{m-6})$  ...  $(2^1, 2^0)$  eines jeden Zweigmetrik-Wortes nacheinander im Halbtakt-Abstand  $T/2$  in die Kaskade getaktet, beginnend mit dem höchstwertigen Bitpaar  $(2^{m-1}, 2^{m-2})$ . Dieses höchstwertige Zweigmetrik-Bitpaar wird aus Nullen bestehen, ebenso auch eines oder mehrere direkt folgende Bitpaare, da der Wert der Zweigmetriken  $ZMi$  kleiner ist als der akkumulierte Wert, den die Pfadmetriken  $PMi$  erreichen können und auf den die Bitbreite  $m$  der Verarbeitung zugeschnitten sein muß. Die Zweigmetrik-Bitpaare durchlaufen die durch die Kaskade gebildete Pipeline mit der Taktfrequenz  $2/T$ . Mit dem Erscheinen des Bits  $2^{m-1}$  (also des MSB) des ersten Zweigmetrik-Wortes werden alle Entscheidungsbits der ersten Kaskadenstufe, also die Bits  $EBPi((m/2)-1)$  und  $EBFi((m/2)-1)$ , auf den Binärwert 1 gestellt und in der Folgezeit auf diesem Wert gehalten. In der Notation der Bitpaare, wie sie in der vorstehenden Beschreibung anhand der Figuren 11 und 12 benutzt wird, lautet die Eingabefolge:  $(m/2)$ -tes Bitpaar;  $((m/2)-1)$ -tes Bitpaar;  $((m/2)-2)$ -tes Bitpaar; ...; erstes Bitpaar. Das heißt, ein " $q$ -tes" Bitpaar enthält die Bits  $2^{2n}$  und  $2^{2n+1}$ , und ein " $(q+1)$ -tes Bitpaar enthält die Bits  $2^{2n+2}$  und  $2^{2n+3}$ , wobei

n von 0 bis m-1 reicht und q von 0 bis  $(m/2)-1$  reicht.

In jedem q-ten Prozessorelemente  $PE_i(q)$  wird im ADD-Block das Bitpaar  $2^{2n}$  und  $2^{2n+1}$  (q-tes Bitpaar) der Zweigmetrik  $ZM_i$  zu der Zahl addiert, die durch die 4 Bits  $PMMA:d(q)$  dargestellt wird. Hierbei handelt es sich um eine redundante Darstellung des q-ten Bitpaares der maximalen Pfadmetrik aus der jeweils zuständigen ACSU-Abteilung. Das Maximum aus den vier neuen Pfadmetriken wird ausgewählt, und für jeden der vier Zweige werden die beiden Entscheidungsbits  $EBP_i(q)$  und  $EBF_i(q)$  gebildet (Fig. 11). Das Zweigmetrik-Bitpaar deckt dabei den Wertebereich  $(0:3) \cdot 2^n$  ab (das Symbol  $*$  ist das Multiplikationszeichen). Die redundante 4-Bit-Darstellung  $PMMA:d(q)$  deckt den Wertebereich  $(0:4) \cdot 2^n$  ab. Der Ausgang des ADD-Blockes deckt den Wertebereich  $(0:7) \cdot 2^n$  ab, wobei das Carry-Bit  $CA_i(q)$  mit dem Wert  $4 \cdot 2^n$  zum vorangehenden Prozessorelement  $PE(q+1)$  weitergereicht wird. Am Eingang des MOD-Blockes liegt somit wieder eine Pfadmetrik-Darstellung für einen Wertebereich  $(0:4) \cdot 2^n$  an, gebildet durch die beiden Summenbits  $SU_{0i}(q)$  und  $SU_{1i}(q)$  und das Carry-Bit  $CA_i(q-1)$  aus dem nachfolgenden Prozessorelement  $PE(q-1)$ .

Der MOD-Block modifiziert diese Zahlendarstellung wie folgt:

Tabelle 7

Eingangsinformation von MOD					Modifizierte Darstellung (Thermometerdarstellung)			
Zahlenwert		Eingangsbits						
dezim.	dual	SU0	SU1	CA	a	b	c	d
0	000	0	0	0	0	0	0	0
1	001	0	0	1	1	0	0	0
		1	0	0				
2	010	0	1	0	1	1	0	0
		1	0	1				
3	011	1	1	0	1	1	1	0
		0	1	1				
4	100	1	1	1	1	1	1	1

Mit der so modifizierten Darstellung ist es möglich, durch die beschriebenen vier 4-ODER-Verknüpfungen im MAX-Block die redundante 4-Bit-Darstellung  $PMMa:d(q)$  des Maximums der insgesamt vier präliminären 4-Bit-Darstellungen  $PMP1a:d(q)$ ,  $PMP2a:d(q)$ ,  $PMP3a:d(q)$  und  $PMP4a:d(q)$  zu gewinnen.

Die modifizierte Darstellung ist eine sogenannte "Thermometer"-Darstellung, die mehr Bits benötigt als die Dualzahl-darstellung. Allgemein besteht eine Thermometerdarstellung einer beliebigen natürlichen Zahl "v" des Zahlenbereiches  $0:u$  aus einem u-Bit-Wort, wobei die Zahl "v" dargestellt wird durch den Binärwert 1 aller v ersten Bits, gezählt ab einem verabredeten Ende (also ab der vorderen oder der hinteren Grenze) des Wortes. Eine Dualzahldarstellung benötigt bekanntlich  $INT[\lg(u+1)]$  Bits (also den ganzzahligen Anteil des Basis-2-Logarithmus von u+1). Die größere Bitbreite der Dualzahldarstellung und der damit verbundene Mehraufwand, insbesondere auch bei der Verteilung der Pfadmetriken, ist jedoch gerechtfertigt wegen der nunmehr einfachen Maximum-Bildung und der einfachen Entscheidungsfindung. Die Maximum-Bildung erfolgt wie bei der bekannten 1-Bit-Architektur mittels ODER-Verknüpfung (Fig. 12).

Die Entscheidungsbits  $EBPi$  und  $EBFi$  werden in den Blöcken MOD und VGL gemäß folgender Vorschrift gebildet:

Tabelle 8

$EBFi(q+1)$	$EBPi(q+1)$	$DIFFi(q)$	$EBFi(q)$	$EBPi(q)$
1	1	$\leq -2$	0	0
		-1	1	0
		$\geq 0$	1	1
1	0	$\leq 2$	0	0
		3	1	0
		4	1	1
0	0	X	0	0

Hierin bedeutet  $DIFFi(q)$  die maximale Differenz zwischen der finalen Pfadmetrik  $PMFi(q)$ , die durch die 4 Bits  $PMFia:d(q)$

dargestellt wird, und den 3 präliminären Pfadmetriken  $PM_j(q)$ , die aus den drei anderen Prozessorelementen  $PE_j(q)$  kommen und jeweils dargestellt werden durch die 4 Bits  $PMP_{ja:d}(q)$ , wobei  $j = 1, 2, 3, 4$  außer  $i$  ist. Das Symbol "X" bedeutet wiederum  
5 "beliebiger Wert".

Hat das finale Entscheidungsbit  $EBFi(q)$  den Wert 0, dann wird dies gewertet als Information zum definitiven Ausschluss des betreffenden Zweiges  $i$ . Die Kombination  $EBFi(q)=1$  und  
10  $EBPi(q)=0$  wird gewertet als Information zu einem voraussichtlichen Ausschluss. Die Kombination  $EBFi(q)=1$  und  $EBPi(q)=1$  besagt, dass der betreffende Zweig vermutlich der Survivor sein wird. Die präliminären und finalen Entscheidungsbits werden beim höchstwertigen Bitpaar (also beim  $(m/2-1)$ -ten  
15 Bitpaar) beginnend berechnet und weitergereicht, bis schließlich am niedrigstwertigen Bitpaar (also beim 0-ten Bitpaar) die eindeutige Entscheidung bereitsteht.

Die Fig. 13 zeigt den kritischen Pfad der neuen 2-Bit-Architektur. Die Zahl der Gatter ist hier genau so groß wie beim  
20 kritischen Pfad (Fig. 10) der bekannten 1-Bit-Architektur. In der 2-Bit-Architektur sind die Gatter jedoch komplexer, was sich ungünstig auf die Laufzeit auswirken könnte. Um dies zu verhindern, ist eine Realisierung der Gatter mittels dynamischer Schaltungstechniken, vorzugsweise mittels der aus [6]  
25 an sich bekannten DOMINO-Logik, besonders vorteilhaft. Bei Implementierung dieser Technik kann die Funktion der Latch-Schaltungen  $L\Phi A$  bzw.  $L\Phi B$ , die in den Figuren 11 und 12 als gesonderte Elemente dargestellt sind, gut in die ADD- und  
30 MOD-Blöcke integriert werden. Eine solche Integration spart zusätzlich Laufzeit.

Bei den "statischen" Logik-Gattern, wie sie insbesondere in integrierten Schaltungen bisher bevorzugt werden, wird die  
35 voll-komplementäre CMOS-Technik angewandt. Hierbei ist das Netzwerk gesteuerter Schaltelemente, welche durch die Eingangsvariablen je nach deren Binärwert durchgeschaltet oder

gesperrt werden, doppelt realisiert, einmal durch MOS-Feldefekttransistoren mit n-leitendem Kanal (N-FETs) und einmal in komplementärer Weise durch MOS-Feldeffekttransistoren mit p-leitendem Kanal (P-FETs). Die beiden Netzwerke sind hintereinander zwischen die Versorgungspotentiale geschaltet, und die Ausgangsgröße wird am Verbindungspunkt abgegriffen. Der Vorteil dieser Anordnung ist, dass Strom nur während der kurzen Zeitspanne fließt, in welcher ein Zustandswechsel an den Eingängen oder am Ausgang stattfindet. Wegen der großen Anzahl der Transistoren, insbesondere bei komplexen Logikfunktionen, ist aber der Platzbedarf solcher Gatter ziemlich groß. Wenn man statt des komplementären P-FET-Schaltnetzes nur einen einzigen P-FET als Pull-Up-Transistor einsetzt, ist der Platzbedarf zwar kleiner, jedoch erfordert der statische Pull-Up-Strom dann viel Leistung und verlangsamt das Pull-Down. Bei dynamischen Logik-Gattern ist das besagte Schaltnetzwerk nur einmal vorgesehen, und zwar unter Verwendung von N-FETs, die weniger Platz beanspruchen als P-FETs. Statt des zweiten Schaltnetzwerkes oder eines Pull-Up-Transistors ist eine Vorlade-, Auswerte- und Halteschaltung vorgesehen, die mit wenigen Transistoren auskommt.

Die Fig. 14 zeigt ein Beispiel dynamischer Schaltungstechnik, speziell für ein Gatter mit einer Logikfunktion gemäß der Booleschen Gleichung:

$$y = [(x1 \cdot x2) + x3] \cdot x4 ,$$

wie es mehrfach in der MOD-Schaltung nach Fig. 12 enthalten ist. Das Symbol  $\cdot$  ist der Operator für UND-Verknüpfung, und das Symbol  $+$  bezeichnet die ODER-Verknüpfung.

Im oberen Teil der Fig. 14 ist dieses "UND-ODER-UND"-Gatter in der üblichen Symboldarstellung gezeichnet, wie sie in Fig. 12 benutzt wird, zusammen mit der nachgeschalteten Latch, bei der es sich z.B. um eine der Latch-Schaltungen LØB in Fig. 12 handeln kann. Der mittlere Teil der Figur zeigt den Aufbau des Gatters in dynamischer Schaltungstechnik. Innerhalb der gestrichelten Umrahmung ist das aus N-FETs N1, N2, N3 und N4

gebildete Schaltnetzwerk für die Logikfunktion gezeichnet. Die Gates dieser N-FETs sind zum Empfang der Eingangsvariablen  $x_1$ ,  $x_2$ ,  $x_3$  und  $x_4$  angeschlossen. Das eine Ende dieses Schaltnetzwerkes liegt am negativeren der beiden Versorgungspotentiale (z.B. Masse), welches das "niedrige" Logikpotential für den Binärwert "0" sei. Das andere Ende ist über die Reihenschaltung eines als N-FET gebildeten Auswertetransistors N5, der durch ein Auswertesignal EVL (evaluate) steuerbar ist, und eines als P-FET gebildeten Vorladetransistors P1, der durch ein Vorladesignal PRC (precharge) steuerbar ist, mit dem positiveren Versorgungspotential verbunden, welches das "hohe" Logikpotential für den Binärwert "1" sei. Des weiteren ist eine Halteschaltung vorgesehen, bestehend aus einem dem Vorladetransistor P1 parallelgeschalteten P-FET P2 und einem Inverter INV, über den der Drain des P-FET P2 auf das Gate rückgekoppelt ist. Die Ausgangsgröße  $y$  wird am Ausgang des Inverters INV abgegriffen.

Im unteren Teil der Fig. 14 ist ein Zeitdiagramm der Signale EVL und PRC und des Ausgangssignals  $y$  dargestellt. Das dynamische Gatter nach Fig. 14 wird wie folgt betrieben: Zum Auswerten der Eingangsbedingung wird zunächst ein Vorladebetrieb realisiert, indem zuerst EVL und kurz danach PRC von hohem auf niedrigen Pegel gebracht werden, so dass P1 leitet und der Knoten K2 zwischen P1 und N5 auf hohem Pegel geht. Hierdurch geht der Ausgang des Inverters INV auf "0". Kurz danach wird PRC wieder auf hohen Pegel gebracht; der Inverter INV behält jedoch seinen Zustand und K2 bleibt hoch, infolge der Rückkopplung über P2. Nach diesem Vorladebetrieb erfolgt die eigentliche Auswertung, indem EVL wieder auf hohen Pegel gebracht wird und N5 somit leitend wird. Erfüllen die Eingangsvariablen die Bedingung

$$[(x_1 \cdot x_2) + x_3] \cdot x_4 = 1,$$

dann ist das Schaltnetzwerk (innerhalb des gestrichelten Rahmens) niederohmig, so dass der Knoten K auf niedrigen Pegel geht und der Inverter eine "1" liefert. Sollte die obige Bedingung nicht erfüllt sein, ist das Schaltnetzwerk hochohmig,

und der Inverterausgang bleibt auf "0".

In der beschriebenen Weise kann jedes beliebige Logik-Gatter als "dynamisches" Gatter unter Verwendung relativ weniger Transistoren konstruiert werden, indem man das Schaltnetzwerk innerhalb des gestrichelten Rahmens gemäß der jeweils gewünschten Logikfunktion ausbildet. Somit lässt sich der Aufwand für die komplexen Gatterschaltungen in den Blöcken MOD und ADD einer erfindungsgemäßen ACSU in vertretbaren Grenzen halten. Ein weiterer Vorteil ist, dass für die Latch-Schaltungen an den Ausgängen der MOD-Blöcke (Figuren 11 und 12) keine eigenen Schaltungen erforderlich sind, denn die Latch-Funktion kann von der ohnehin vorhandenen Halteschaltung (Inverter INV und Transistor P2) am Ausgang jedes dynamischen Gatters übernommen werden. Hierzu können die Taktsignale für die Halbtakte  $\Phi A$  und  $\Phi B$  so geformt werden, dass sie die Rolle der Auswertesignale EVL an den dynamischen Gattern übernehmen.

Die dynamischen Gatter können in vorteilhafter Ausgestaltung der Erfindung speziell gemäß der sogenannten DOMINO-Technik ausgebildet sein, wie sie detailliert in [6] beschrieben ist. Hierbei erfolgt das Vorladen des Ausgangs auf hohen Pegel, während der zum niedrigen Potential führende Weg geöffnet wird, und das Vorladen wird gestoppt, während der Weg zum niedrigen Potential aktiviert wird.

Die vorstehende Beschreibung anhand der Figuren 11 bis 14 betrifft nur ein Ausführungsbeispiel der Erfindung, betreffend einen Radix-4-Trellis. Das Prinzip der Erfindung ist jedoch nicht beschränkt auf einen solchen Trellis, sondern kann allgemein auf alle denkbaren Radix- $2^x$ -Trellisse mit  $x \geq 1$  angewendet werden. Hierbei ist die Anzahl der Zweigmetriken, die für jeden Trelliszustand verarbeitet werden müssen, und die Anzahl der zu vergleichenden akkumulierten Pfadmetriken natürlich entsprechend niedriger oder höher. Auch kann die Anzahl der Trelliszustände größer oder kleiner als 8 sein;

derzeit wird in Praxis mit 16 oder 32 (oder sogar noch mehr) Trelliszuständen gearbeitet.

Auch ist die Erfindung nicht beschränkt auf die Einrichtung einer 2-Bit-Architektur, wie sie vorstehend detailliert als Beispiel beschrieben wurde. Die Prozessorelemente der Kaskade können auch so ausgestaltet werden, dass sie  $p > 2$  Zweigmetrik-Bits simultan verarbeiten. Eine solche Ausgestaltung liegt natürlich ebenfalls im Bereich der Erfindung.

Ferner sei erwähnt, dass die in der Fig. 12 dargestellten speziellen Ausbildungen der Blöcke ADD, MOD, VGL und MAX nur Beispiele bzw. bevorzugte Ausführungsformen sind. Ausschlaggebend sind die von diesen Blöcken zu realisierenden Logikfunktionen, die auch durch andere als die gezeigten Beispielschaltungen verwirklicht werden können.

Die vorangehend als Beispiel beschriebene ACSU arbeitet mit Maximum-Auswahl. Wird zur Ermittlung der Zweigmetriken in der BMU eine Abstandsfunktion verwendet, die eine Minimum-Auswahl für die Survivoren erfordert, müssen manche Logikfunktionen in der ACSU invertiert werden. Ein Fachmann wird ohne weiteres in der Lage sein, die hierzu notwendigen Modifikationen vorzunehmen.



## Patentansprüche

1. Viterbi-Decoder zum Decodieren eines Faltungscodes auf der Basis eines Radix- $2^x$ -Trellis mit  $2^z$  Zuständen, wobei  $x$  und  $z$  ganze Zahlen  $\geq 1$  sind, enthaltend:

eine erste Berechnungseinheit (BMU), die aus periodisch aufeinander folgenden Mehrbit-Wörtern einer Eingangssequenz für jeden Trelliszustand die  $2^x$  Zweigmetriken (ZMi) als Mehrbit-Wörter berechnet;

eine zweite Berechnungseinheit (ACSU), die für jeden Trelliszustand in jeder Taktperiode die jeweils zugeordneten  $2^x$  Zweigmetriken (ZMi) mit bisher akkumulierten  $2^x$  Pfadmetriken (PMM) aus  $2^x$  Vorgänger-Trelliszuständen addiert und die in einem vorgewählten Sinne extreme der so ermittelten  $2^x$  Pfadmetriken (PMi) als neue akkumulierte Pfadmetrik (PPM) für den Additionsvorgang der nächsten Periode auswählt;

eine dritte Berechnungseinheit (SMU) zum Speichern der aus der zweiten Berechnungseinheit (ACSU) gewonnenen Informationen über die Identität (i) der Pfadmetriken (PMi), die in den aufeinander folgenden Taktperioden in der zweiten Berechnungseinheit ausgewählt werden,

wobei die zweite Berechnungseinheit (ACSU) für jeden Trelliszustand enthält: eine Anzahl  $2^x$  paralleler Kaskaden von Prozessorelementen (PE), die hintereinander geschaltet sind für eine Pipeline-Verarbeitung der Bits der Zweigmetriken und der extremen Pfadmetriken (PPM), fortschreitend mit abnehmendem Stellenwert der Bits, und jeweils eine Extremwert-Auswahleinrichtung (MAX) für alle Prozessorelemente gleicher Ordnungszahl (q) innerhalb der Pipeline;

d a d u r c h g e k e n n z e i c h n e t ,

dass die Anzahl der Prozessorelemente (PE) in jeder der Kaskaden kleiner ist als die Anzahl  $m$  der Bits, die für die Dualzahldarstellung des Wertebereiches der Pfadmetriken (PM) verwendet werden, indem jede der Kaskaden jeweils eine dem ganzzahligen Anteil von  $m/p$  entsprechende Anzahl  $\text{INT}(m/p)$  von Prozessorelementen ( $\text{PE}1:m/p$ ) enthält, wobei  $p$  eine ganze Zahl mindestens gleich 2 und kleiner als  $m$  ist und wobei jedes

dieser  $\text{INT}(m/p)$  Prozessorelemente innerhalb der Kaskade genau einem von  $\text{INT}(m/p)$  aufeinander folgenden disjunkten  $p$ -Bit-Gruppen der  $m$  Bits zugeordnet ist.

- 5    2.    Viterbi-Decoder nach Anspruch 1, gekennzeichnet durch derartige Bemessung der Zahlen  $m$  und  $p$ , dass  $m/p$  ganzzahlig ist, so dass jede Kaskade aus insgesamt  $m/p$  Prozessorelementen besteht, deren jedem eine  $p$ -Bit-Gruppe zugeordnet ist.
- 10   3.    Viterbi-Decoder nach Anspruch 1, dadurch gekennzeichnet, dass  $m/p$  nicht-ganzzahlig ist und dass jede Kaskade zusätzlich zu den  $\text{INT}(m/p)$  Prozessorelementen ein weiteres Prozessorelement enthält, das der  $r$ -Bit-Gruppe zugeordnet ist, die als Rest nach Abzug der  $\text{INT}(m/p)$   $p$ -Bit-Gruppen bleibt.
- 15   4.    Viterbi-Decoder nach Anspruch 3, dadurch gekennzeichnet, dass das weitere Prozessorelement den  $r$  höchstwertigen Bits oder den  $r$  niedrigstwertigen Bits der  $m$  Bits zugeordnet ist.
- 20   5.    Viterbi-Decoder nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet dass jedes Prozessorelement ( $\text{PE}_i$ ) folgendes enthält:
  - eine Addiereinrichtung (ADD) zum Bilden der als Dualzahl dargestellten Summe zweier Zahlen, deren eine dargestellt ist als Dualzahl durch die zugeordnete Gruppe der Zweigmetrik-Bits und deren andere eine redundante Darstellung ( $\text{PMMa:d}$ ) der zugeordneten Gruppe der Bits der bisher akkumulierten extremen Pfadmetrik ist;
  - eine Modifiziereinrichtung (MOD), die als Eingangsgrößen
- 30   die  $p$  niedrigstwertigen Bits ( $\text{SUI-0}$ ,  $\text{SUI-1}$ ) der besagten Summe und das Carry-Bit ( $\text{CA}_i$ ) aus der Addiereinrichtung des in der Kaskade nachfolgenden Prozessorelementes (Ordnungszahl  $q-1$ ) und ein präliminäres und ein finales Entscheidungsbit ( $\text{EBP}$ ,  $\text{EBF}$ ) von dem in der Kaskade vorhergehenden Prozessorelement (Ordnungszahl  $q+1$ ) empfängt, um eine redundante Darstellung ( $\text{PMPia:d}$ ) der zugeordneten Gruppe der Bits einer präliminären Pfadmetrik (PMP) und eine redundante Darstellung
- 35

(PMFia:d) der zugeordneten Gruppe der Bits einer finalen Pfadmetrik (PMF) zu liefern;

5 eine Vergleicheinrichtung (VGL), die durch Verknüpfung der finalen Pfadmetrik-Darstellung (PMFia:d) aus dem betreffenden Prozessorelement (PEi) mit den  $2^x-1$  präliminären Pfadmetrik-Darstellungen (PMFja:d) aus den anderen Prozessorelementen (PEj) gleicher Ordnungszahl (q) das präliminäre Entscheidungsbit (EBPi) und das finale Entscheidungsbit (EBFi) für das nachfolgende Prozessorelement (Ordnungszahl q-1)  
10 bildet.

6. Viterbi-Decoder nach Anspruch 5, dadurch gekennzeichnet dass zwischen den Ausgängen der Addiereinrichtung (ADD) und den Eingängen der Modifiziereinrichtung (MOD) ein erstes  
15 Latchregister (LΦA oder LΦB) vorgesehen ist und dass an den Ausgängen der Modifiziereinrichtung (MOD) ein zweites Latchregister (LΦB oder LΦA) vorgesehen ist und dass die beiden Latchregister um eine halbe Periode der Taktrate versetzt getaktet sind.

20

7. Viterbi-Decoder nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass die extremen Pfadmetriken (PMM) die jeweils maximalen Pfadmetriken sind.

25

8. Viterbi-Decoder nach einem der Ansprüche 5 bis 7, dadurch gekennzeichnet,

30 dass die redundanten Darstellungen Thermometerdarstellungen sind, jeweils bestehend aus einem u-Bit-Wort, wobei u der höchste Wert der darzustellenden natürlichen Zahl ist und der tatsächliche Wert v dargestellt ist durch einen selben vorgewählten Binärwert genau aller v ersten Bits, gezählt ab einem vorgewählten Ende des Wortes,

35 und dass jede Extremwert-Auswahleinrichtung (MAX) eine Anzahl u logischer ODER-Gatter enthält, deren jedes die  $2^x$  Bits gleicher Ordnungszahl aus den Thermometerdarstellungen derjenigen präliminären Pfadmetrikbits (PMPi) empfängt, die

der betreffenden Gruppe der Zweigmetrik-Bits zugeordnet sind, so dass jedes der besagten ODER-Gatter an seinem Ausgang eines der  $u$  Bits der Thermometerdarstellung (PPMa:d) der zugeordneten Bit-Gruppe der bisher akkumulierten extremen Pfadmetrik liefert.

9. Viterbi-Decoder nach Anspruch 8, dadurch gekennzeichnet, dass  $p = 2$  ist und dass  $u = 4$  ist.

10. Viterbi-Decoder nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, dass  $p = 2$  ist.

11. Viterbi-Decoder nach einem der Ansprüche 1 bis 10, dadurch gekennzeichnet, dass  $x = 2$  ist.

12. Viterbi-Decoder nach einem der Ansprüche 5 bis 11, dadurch gekennzeichnet,

dass die Addiereinrichtung (ADD), die Modifiziereinrichtung (MOD) und die Vergleicheinrichtung (VGL) jedes Prozes-  
sorelementes logische Gatter enthalten,

und dass alle logischen Gatter in der zweiten Berechnungseinheit (ACSU) dynamische Schaltungen sind, vorzugsweise in DOMINO-Technologie.

13. Viterbi-Decoder nach den Ansprüchen 6 und 12, dadurch gekennzeichnet, dass die Funktionen der Latchregister ( $L\Phi A$ ,  $L\Phi B$ ) in den dynamischen Gattern der Addiereinrichtung (ADD) und der Modifiziereinrichtung (MOD) implementiert sind.

## Zusammenfassung

## Viterbi-Decoder

5

Beschrieben wird ein Viterbi-Decoder mit einer Berechnungseinheit (ACSU) für einen Radix- $2^x$ -Trellis, die für jeden Trelliszustand die jeweils zugeordneten  $2^x$  Zweigmetriken (ZMi) mit bisher akkumulierten  $2^x$  Pfadmetriken (PMM) aus  $2^x$  Vorgänger-Trelliszuständen addiert und die extreme der so ermittelten  $2^x$  Pfadmetriken (PMi) als neue akkumulierte Pfadmetrik (PMM) für den Additionsvorgang der nächsten Periode auswählt, wobei besagte Berechnungseinheit enthält:

10 für jeden Trelliszustand  $2^x$  parallele Kaskaden von Prozessorelementen (PE) zur Pipeline-Verarbeitung der Bits der Zweigmetriken und der Pfadmetriken (PMM), und für jeden Trelliszustand jeweils eine Extremwert-Auswahleinrichtung (MAX) für alle Prozessorelemente gleicher Ordnungszahl (q)

15 innerhalb der Pipeline. Erfindungsgemäß ist die Anzahl der Prozessorelemente (PE) in jeder der Kaskaden kleiner als die Anzahl m der Bits, die für die Dualzahldarstellung des Wertebereiches der Pfadmetriken (PM) verwendet werden. Hierzu sind Prozessorelemente in jeder Kaskade so ausgebildet, dass sie

20 jeweils disjunkte Gruppen von  $p \geq 2$  Bits zusammenhängend verarbeiten.

25

(Fig. 11)

Figur für die Zusammenfassung

ACSU-Abteilung A aus Fig. 7  
(speziell: Prozessorelemente für Zweigmetrik ZM4)

2-Bit-Prozessorelement PE4(q+1)  
für Bits (2n+2) und (2n+3)

2-Bit-Prozessorelement PE4(q)  
für Bits (2n) und (2n+1)

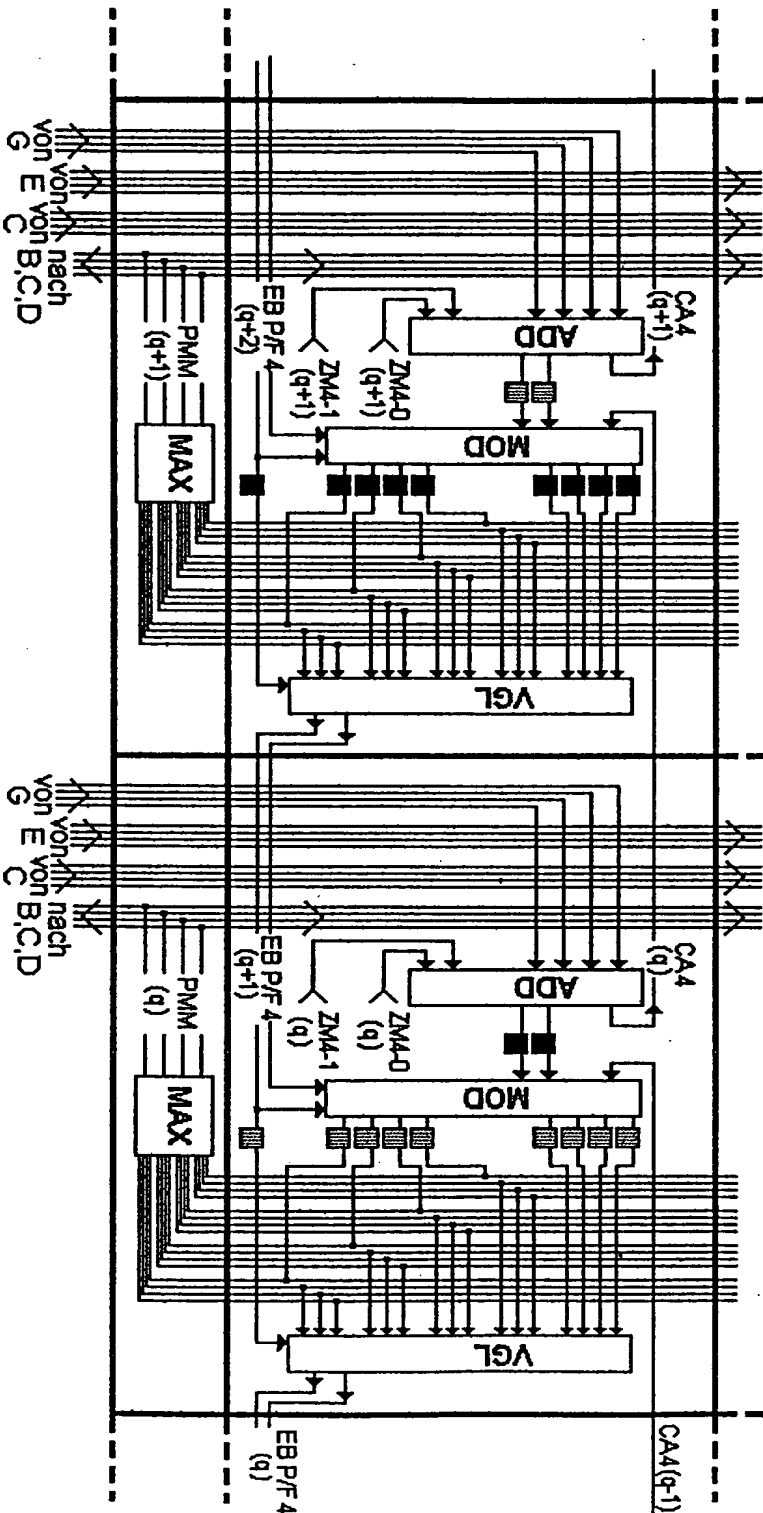


FIG. 11

# Radix-2-Trellis mit vier Zuständen, digitale Übertragung

Beispiel: Zweigmetrik ZM = [ 2 - Hammingdistanz (s, e) ]

Original-Sequenz	0	1	1	0	...
Sende-Sequenz	00	11	01	01	
Empfangs-Sequenz	00	11	01	01	

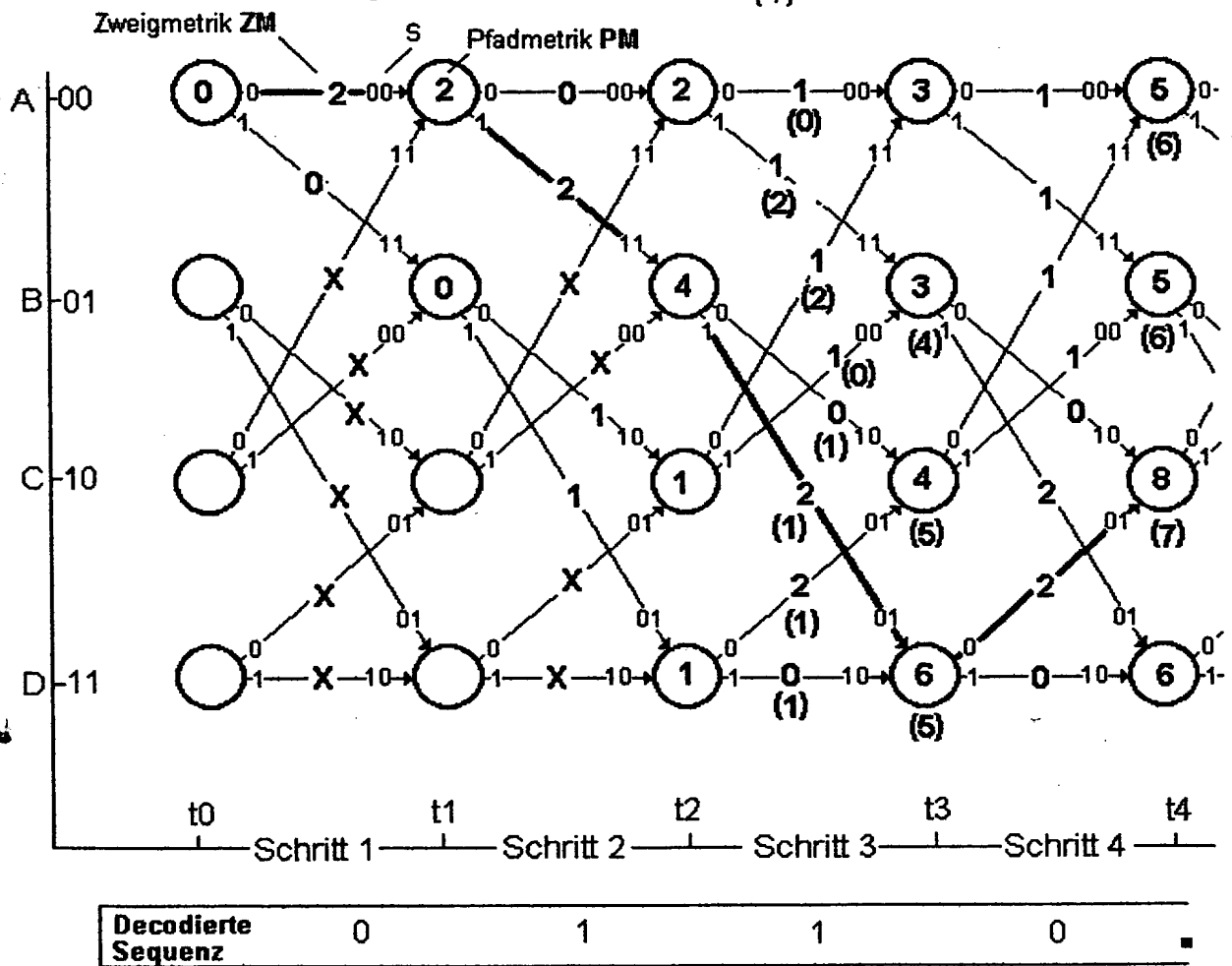


FIG. 1

# Radix-2-Trellis mit vier Zuständen, analoge Übertragung

Beispiel: Zweigmetrik  $ZM = 3 - |s - e|$

Original-Sequenz	0	1	1	0	■	■
Ausgabe-Sequenz	00	11	01	01		
Sende-Sequenz	0	3	1	1		
Empfangs-Sequenz	0,2	2,7	1,6	1,1		

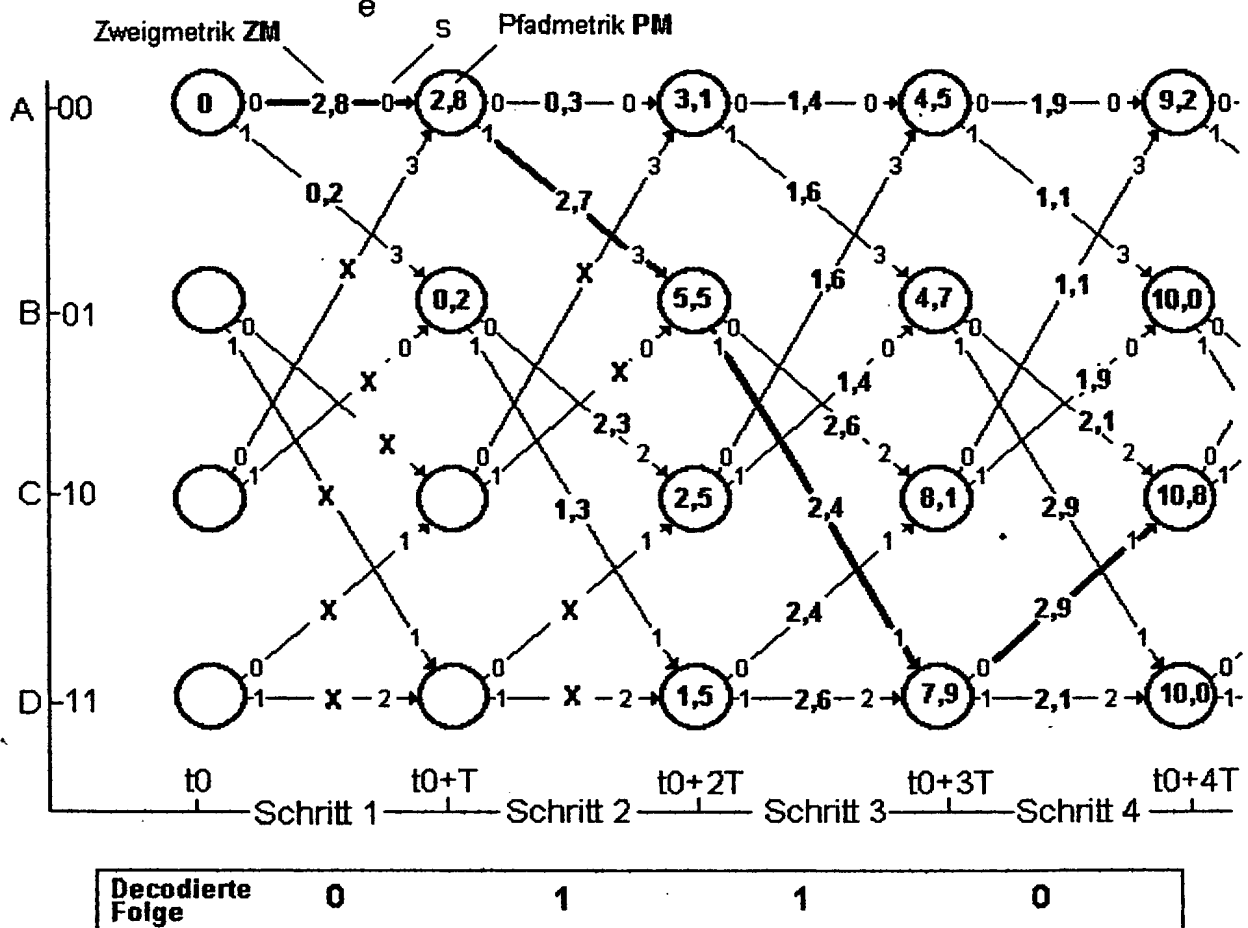


FIG. 2



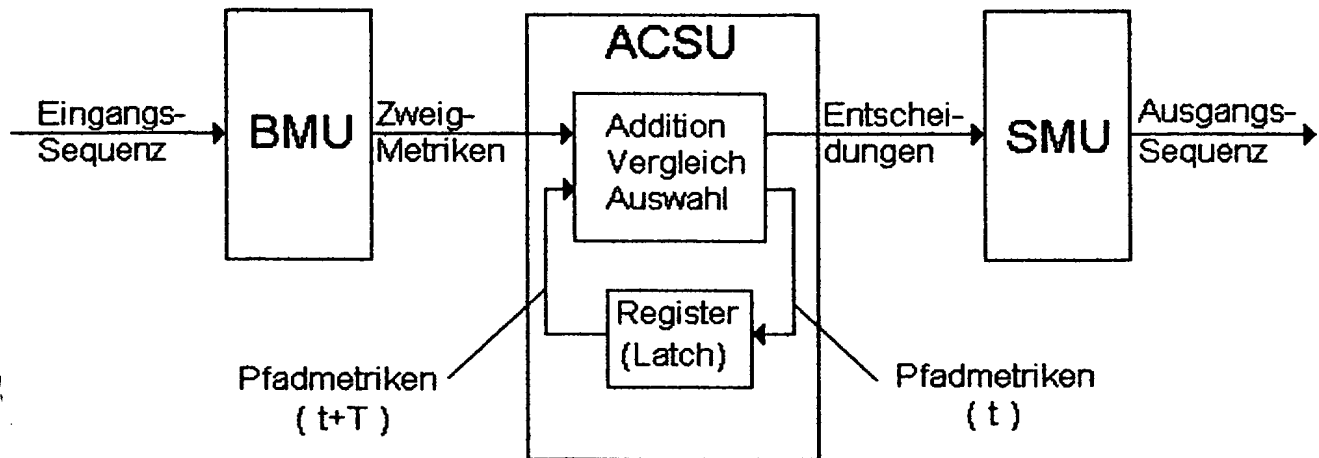


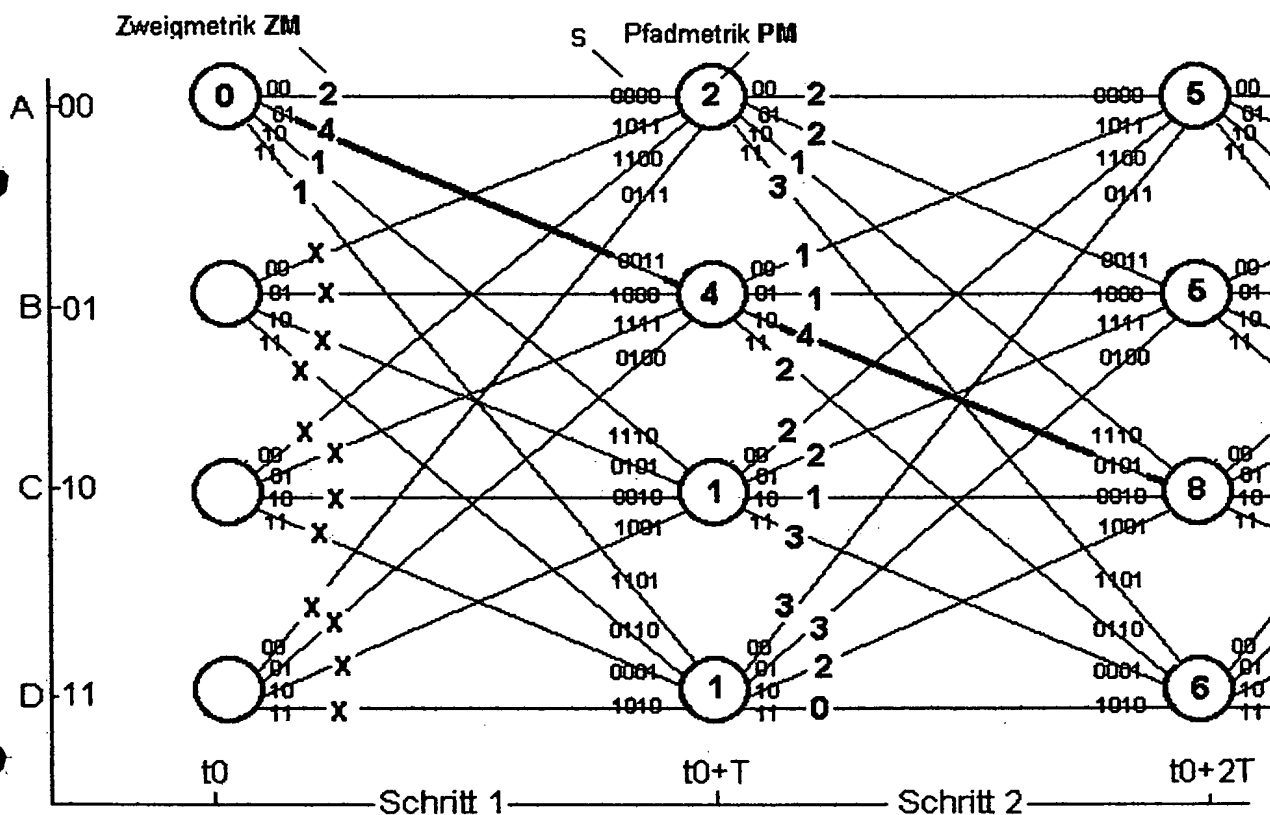
FIG. 3

# Radix-4-Trellis mit vier Zuständen, digitale Übertragung

Beispiel: Zweigmetrik ZM = [4 - Hammingdistanz (s, e)]

Original-Sequenz	0	1	1	0	...
Ausgabe-Sequenz	00	11	01	01	
Empfangs-Sequenz	00	11	01	01	

e



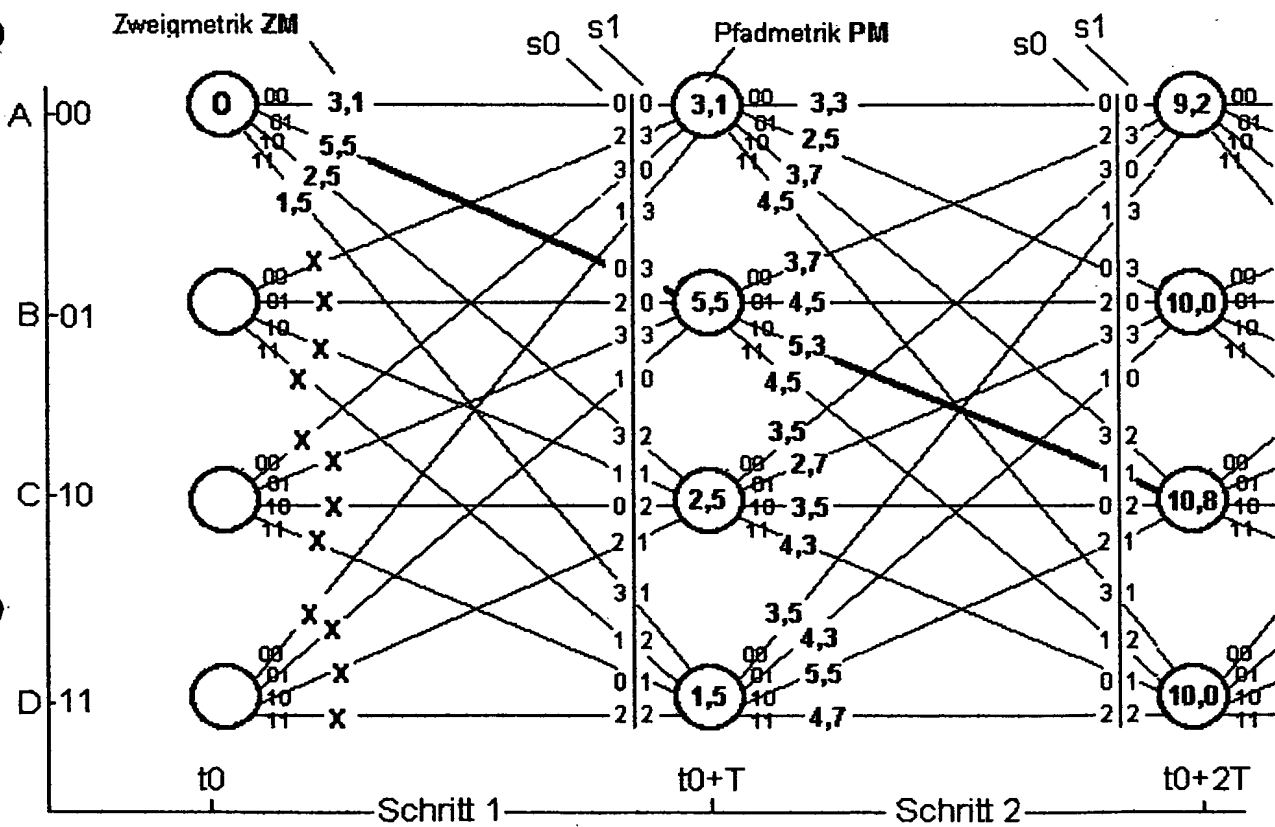
Decodierte Sequenz	0	1	1	0	...
--------------------	---	---	---	---	-----

FIG. 4

# Radix-4- Trellis mit vier Zuständen, analoge Übertragung

Beispiel: Zweigmetrik  $ZM = 6 - (|e0 - s0| + |e1 - s1|)$

Original-Sequenz	0	1	1	0	...
Ausgabe-Sequenz	00	11	01	01	
Sende-Sequenz	0	3	1	1	
Empfangs-Sequenz	0,2	2,7	1,6	1,1	
	e0	e1	e0	e1	



Decodierte Folge: 0 1 1 0

FIG. 5

## Radix-4-Trellis mit acht Zuständen

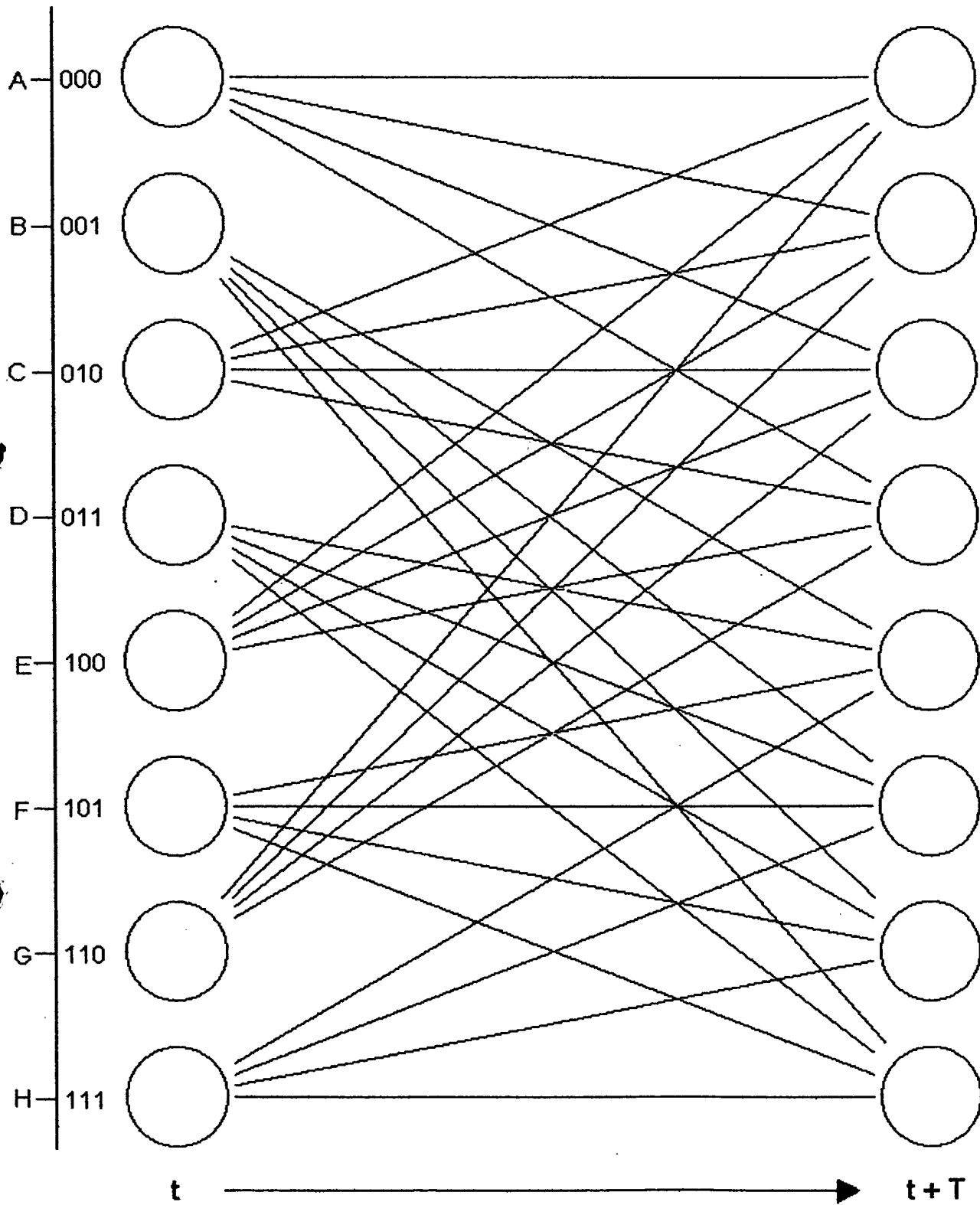


FIG. 6

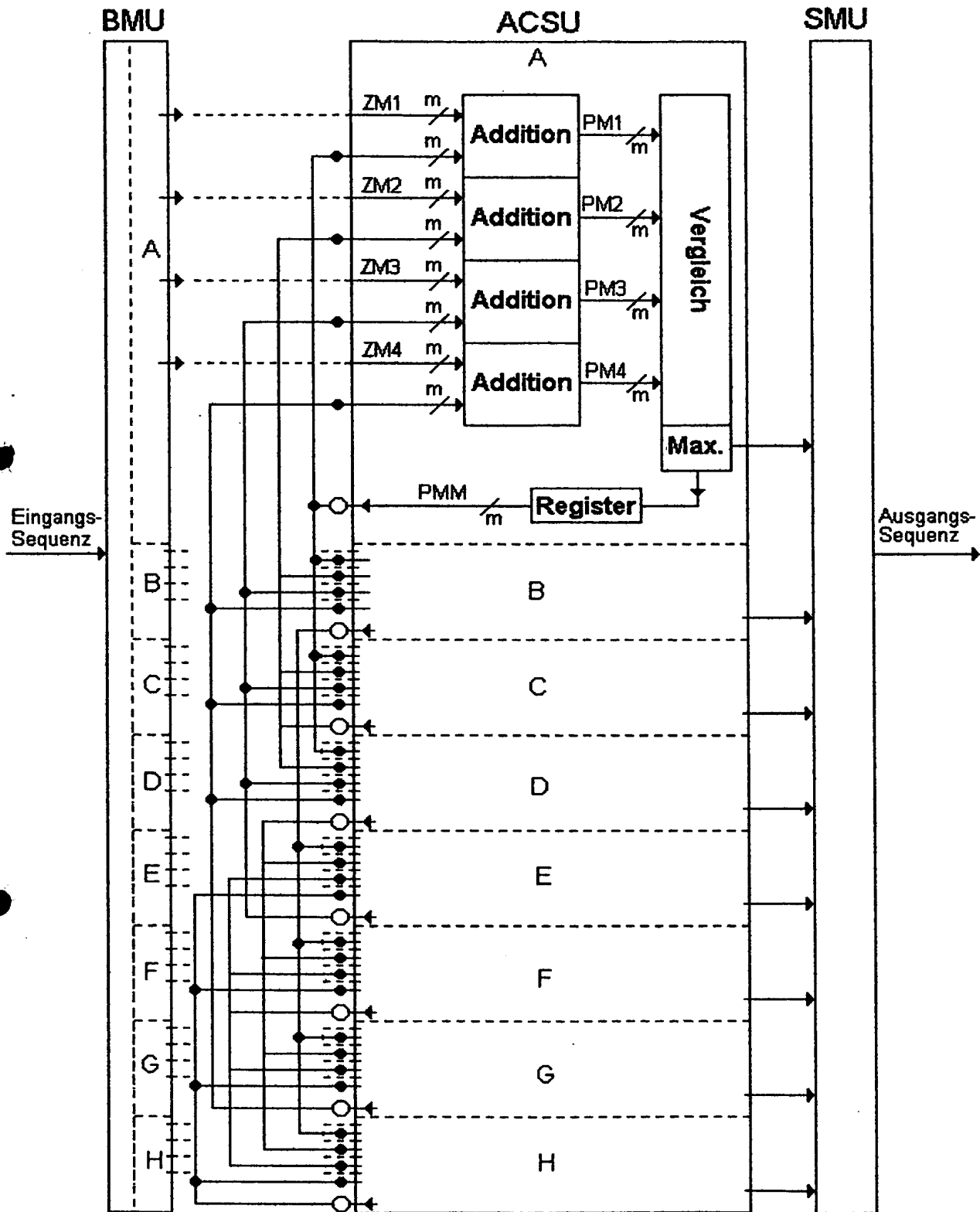
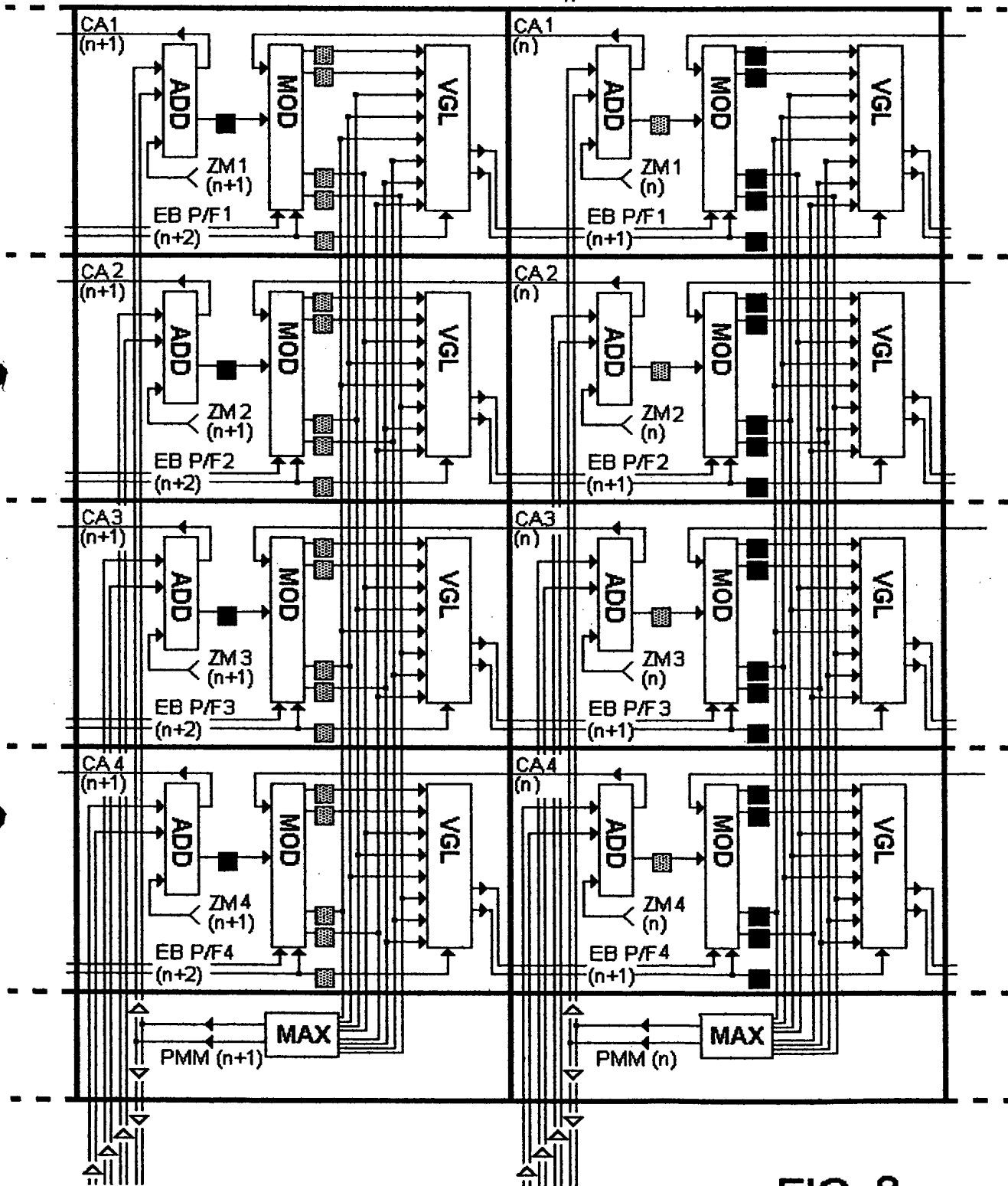


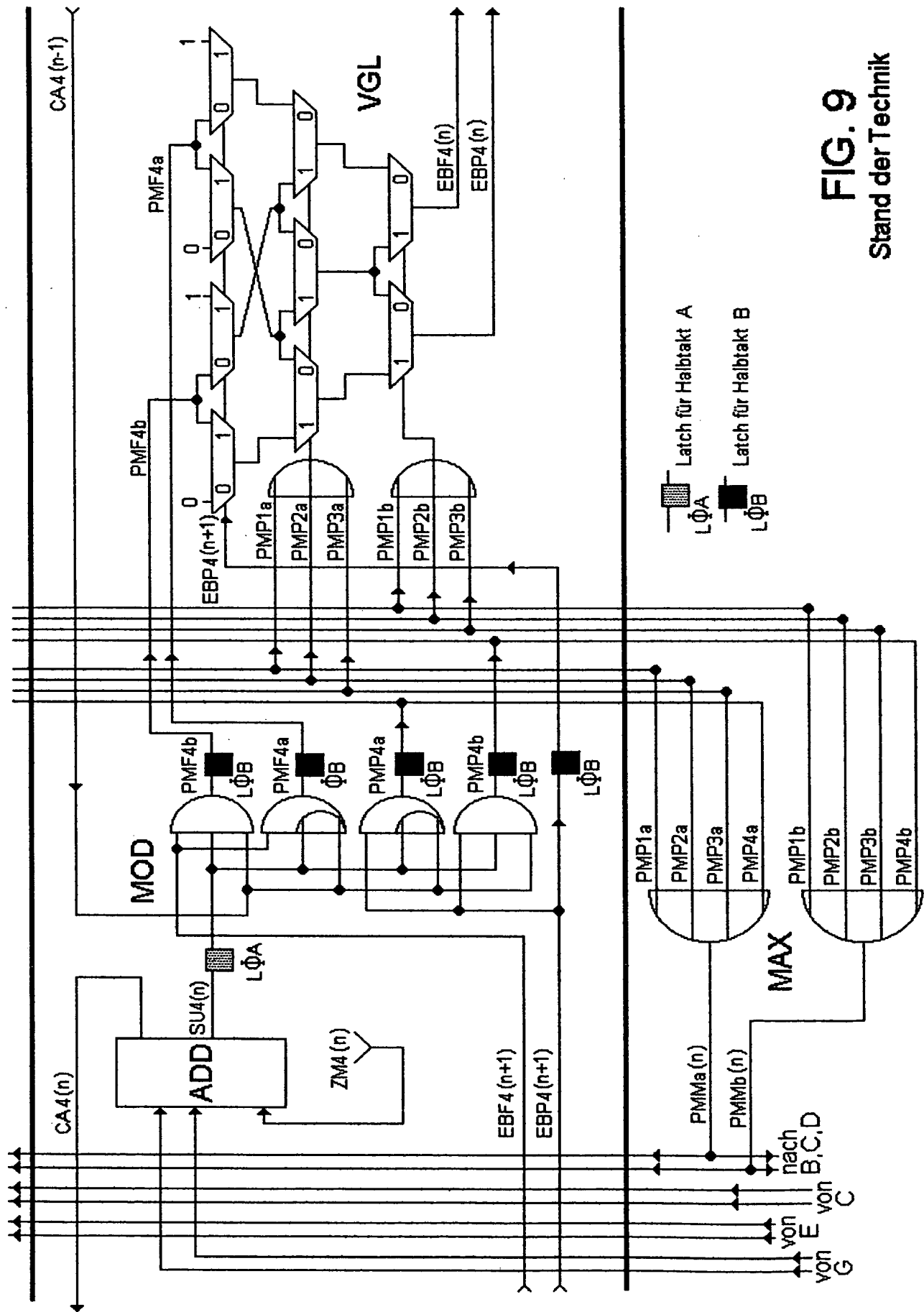
FIG. 7

## ACSU-Abteilung A aus Fig. 7

← 1-Bit-Prozessorelemente PE1:4 für Bit (n+1) → ← 1-Bit-Prozessorelemente PE1:4 für Bit (n) →



**FIG. 8**  
Stand der Technik



**FIG. 9**  
Stand der Technik

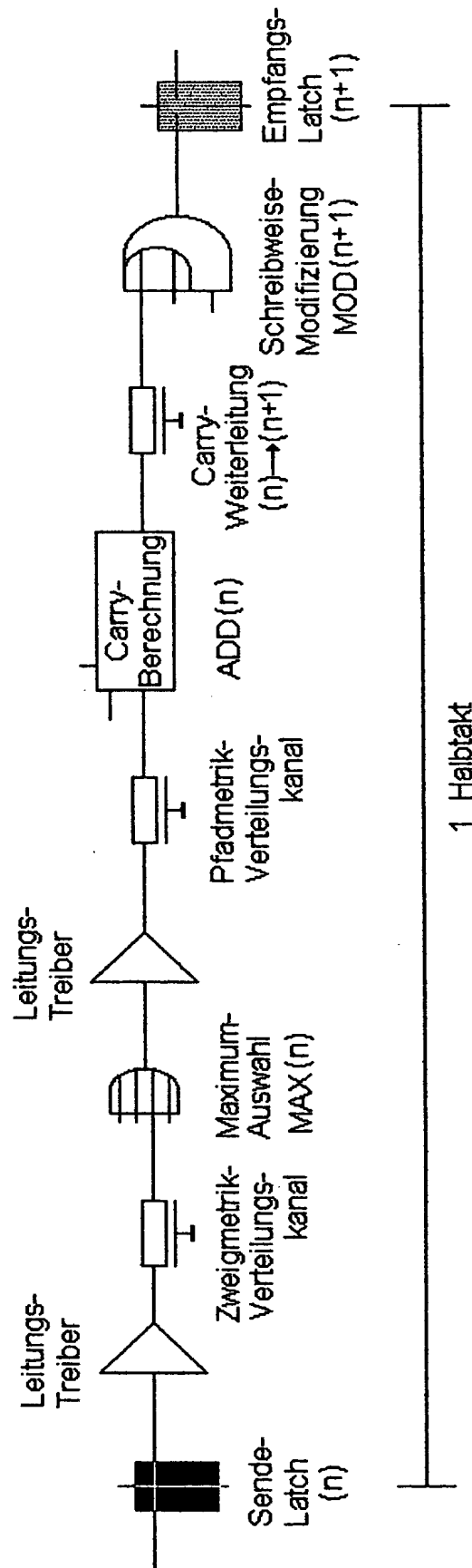


FIG. 10

Stand der Technik



# ACSU-Abteilung A aus Fig. 7

(speziell: Prozessorelemente für Zweigmetrik ZM4)

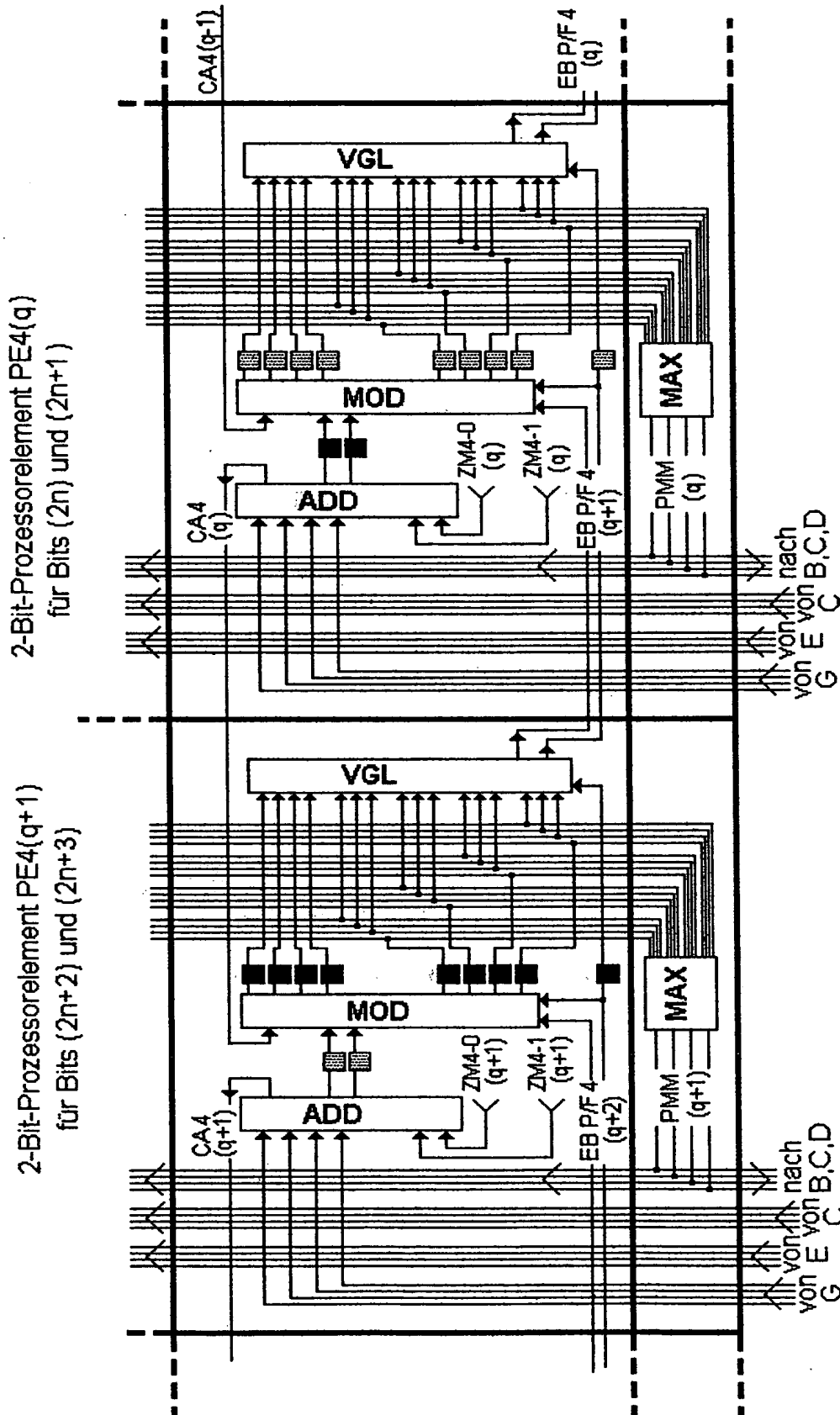


FIG. 11

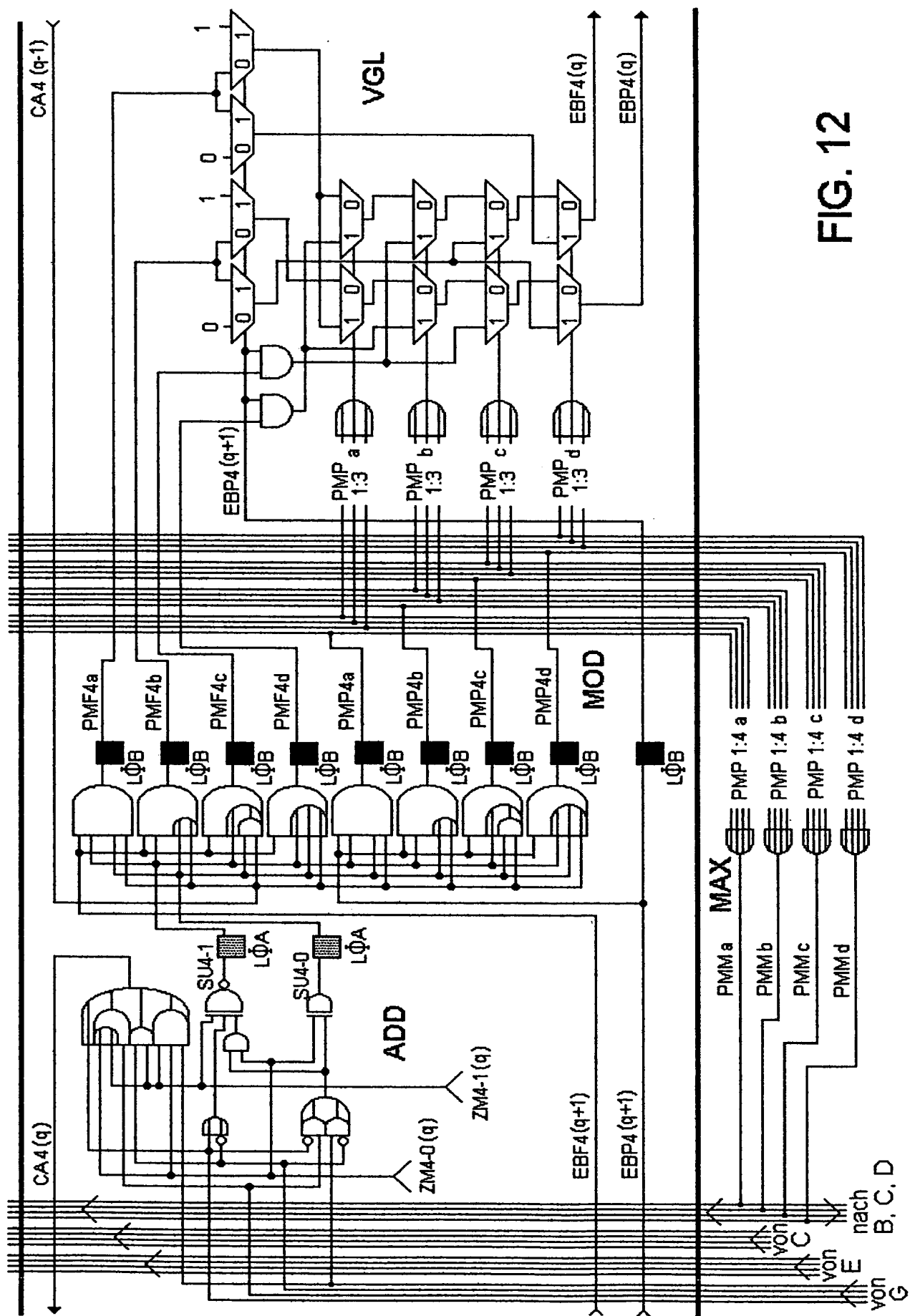


FIG. 12

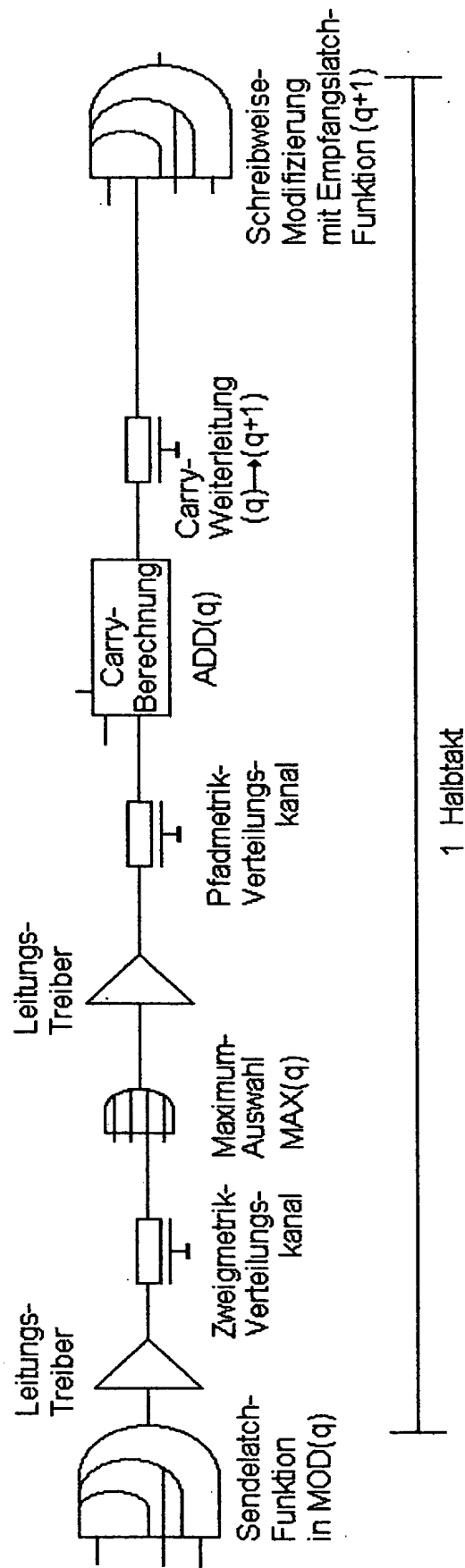


FIG. 13

14/14

$$y = [(x1 \cdot x2) + x3] \cdot x4$$

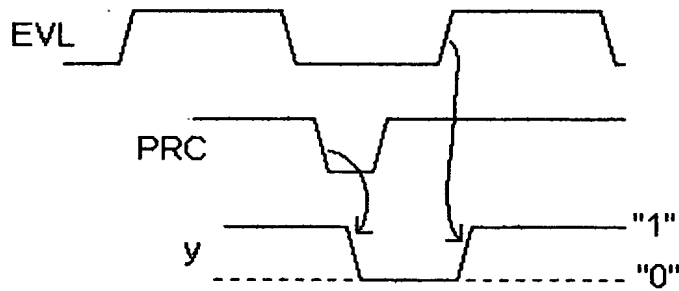
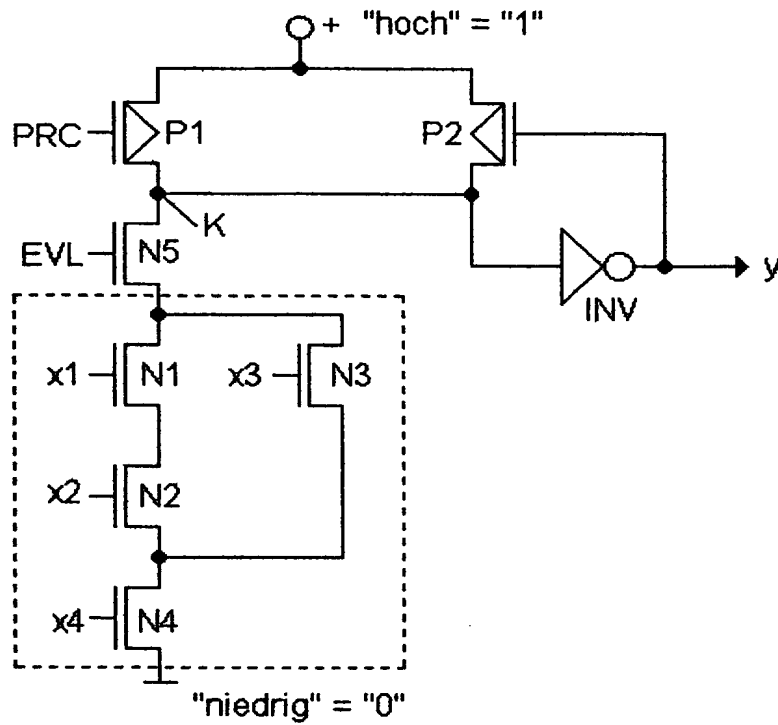
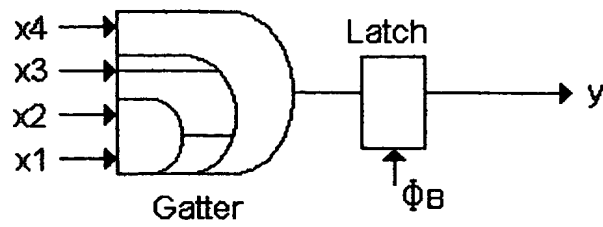


Fig. 14